Making Models

Vom Selbermachen stofflich-digitaler Artefakte als Modellbildung

Dem Fachbereich 3 Mathematik und Informatik der
Universität Bremen
zur Erlangung des akademischen Grades eines
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
eingereichte Dissertation

von
Eva-Sophie Katterfeldt

am 27. November 2014

Gutachter_innen:

1. Gutachterin:
Prof. Dr.-Ing. Heidi Schelhowe
Universität Bremen
Fachbereich 3 Mathematik und Informatik

2. Gutachter:
Prof. Dr.-Ing. Friedrich-Wilhelm Bruns
Universität Bremen
Fachbereich 3 Mathematik und Informatik

Datum des Promotionskolloquiums:

02. März 2015

Für meinen Vater Dr.-Ing. Harald Katterfeldt †

Danksagung

Während der Zeit, die diese Arbeit in Anspruch genommen hat, habe ich von unzähligen Menschen auf vielfältige Weise Unterstützung erfahren. Dafür möchte ich mich bedanken.

Besonderer Dank gilt meinen Betreuer_innen Heidi Schelhowe und Willi Bruns für ihren Rat und die Freiheiten, die sie mir bei Themenwahl und Umsetzung gewährt haben. Meinen aktuellen und ehemaligen Kolleg_innen bei dimeb danke ich herzlichst für fachlichen Austausch, Beistand und Unterstützung, allen voran Nadine Dittert sowie Iris Bockermann, Ingrid Bode, Jennifer Boldt, Corinne Büching, Simon Engelbertz, Dennis Krannich, Roger Meintjes, Milena Reichel, Bernd Robben, Julia Walter-Herrmann, Sabrina Wilske und Anja Zeising. Dank gilt auch den Teilnehmenden und Tutor_innen der beteiligten TechKreativ-Workshops. Ohne sie wären die Untersuchungen dieser Arbeit nicht möglich gewesen.

Danke an die Korrekturleser_innen für ihren fleißigen Einsatz. Nicht zuletzt danke ich meiner Familie für Geduld und Verständnis – und ganz besonders Andreas für seine unermüdliche und klaglose Anteilnahme.

Zusammenfassung

In den vergangenen Jahren ist eine *Makerbewegung* entstanden, die sich dadurch kennzeichnet, dass das Selbermachen von Dingen untrennbar mit Digitalen Medien verbunden ist. Zum einen vernetzen sich weltweit Amateur_innen, um Dokumentationen ihrer Projekte zu tauschen und voneinander zu lernen. Zum anderen erstellen sie selbständig Digitale Medien als stofflich-digitale Artefakte. Um diese anderen zur Nachahmung zur Verfügung zu stellen oder um angefangene Projekte zu einem späteren Zeitpunkt wieder aufzunehmen, bedarf es Projektdokumentationen, die die wesentlichen Eigenschaften des zu konstruierenden Artefakts abbilden. Es stellt sich die Frage, wie junge angehende Maker_innen unterstützt werden können ihre stofflich-digitalen Artefakte während des Konstruierens umfassend und nachvollziehbar abzubilden.

Diese Arbeit entwickelt einen modelltheoretischen Ansatz, um das Selbermachen stofflich-digitaler Artefakte – *Making* – als *Modellbildung* zu beschreiben. Ausgehend von einem allgemeinen Modellverständnis greift sie ähnliche Konzepte aus der Informatik auf, ergänzt sie um alternative unformale Herangehensweisen aus dem HCI-Kontext, überträgt sie auf Making und leitet Anforderungen an Modellierungstools ab. Eine inhaltliche Modellanalyse und die Evaluation neu entwickelter Tools zeigen, dass diverse Bildmodelle als auch Visueller Programmcode aussagekräftige und praxisnahe Modellformate darstellen, mit denen sich unformale und abstrakte Eigenschaften für und von Laien nachvollziehbar darstellen lassen. Es wird empfohlen, Visuelle Programmierumgebungen zu einfachen Dokumentationstools zu erweitern, um am Vorgehen junger Maker anzuknüpfen.

Abstract

In recent years a 'maker movement' that is characterized by linking DIY to digital media has appeared. In this context amateurs create and construct digital physical artefacts themselves, and share documentation of their projects online to learn from each other. Distributing on-going projects to other 'makers' presupposes that essential characteristics of the artefact under construction are documented. This raises the question of how young amateurs can be supported to document their physical digital artefacts.

This thesis develops an approach based on model theory to define the making of digital physical artefacts as modelling processes. Starting from a general model definition, corresponding modelling concepts from computer science are taken up, expanded by alternative un-formal approaches from HCI and applied to 'making'. Requirements for modelling software tools are derived and implemented. A content analysis of makers' models and the evaluation of the new tools points towards the appropriateness of graphical models and unveils benefits of visual programming code for documentation. They are meaningful and practical model formats to outline un-formal and abstract characteristics comprehensibly. It is proposed to extend visual programming environments into simple documentation tools to relate to young makers' construction activities.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Perspektive auf informatische Systeme als Modelle hat meine Forschung begleitet, seit ich mich mit diesem Thema in meiner Master Thesis[1] 2007 befasst habe. Modelle sind mir in meiner Tätigkeit bei dimeb weiterhin begegnet – insbesondere in TechKreativ-Technologieworkshops – und gaben immer wieder Anlass zu Überlegungen, welche Modelle dabei entstehen und wie Modelldokumentationen für Reflexion und Kommunikation genutzt werden können.

Seit ein paar Jahren hat das Selbermachen von Dingen in Verbindung mit Digitalen Medien weltweit Aufmerksamkeit bekommen – bezeichnet mit dem verheißungsvollen Namen „maker movement“ (Dougherty, 2012, S. 11). Die TechKreativ-Workshops[2], die bei dimeb bereits seit einem Jahrzehnt zu Medienbildungszwecken durchgeführt werden, und die damit verbundene Forschung haben so einen weiteren Kontext und ein neues Wirkungsfeld bekommen.

Diese Entwicklung ist untrennbar mit Digitalen Medien verbunden: Erst durch Web2.0-Technologien und darauf basierenden Online-Communitys konnte eine Makerbewegung entstehen. Sie existiert dadurch, dass Menschen ihre DIY-Projekte dokumentieren, veröffentlichen und sich darüber austauschen. Durch ihre Veröffentlichungen partizipieren diese ‚Maker‘ und ‚Makerinnen‘ an der Bewegung und tragen aktiv zu ihrem Fortbestehen bei. Auch sind Digitale Medien Zielobjekte von DIY-Projekten – z.B. als stofflich-digitale Artefakte, die mit Hilfe von Arduino-Mikrocontrollern gebaut werden. Dabei handelt es sich um komplexe informatische Systeme. Sie zu erstellen erfordert ein Formalisieren unformal wahrgenommener Wirklichkeit. So muss ein erdachtes interaktives Artefakt in ein Hardwarekonstrukt und in Programmcode abstrahiert werden.

In diesem Zusammenhang bekommt das Dokumentieren von Konstruktionsprozessen – bzw. der dabei entstehenden Modelle – eine neue Relevanz, die über das

[1] Diese hatte zum Ziel, Kinder beim ‚Modellieren‘ interaktiver Objekte zu unterstützen durch ein allmähliches Abstrahieren der Projektidee (siehe auch Abschnitt 5.3.1 in dieser Arbeit).

[2] `http://www.techkreativ.de` (aufgerufen am 07.01.2014), siehe auch Abschnitt 2.4.2.

Lernen in Technologieworkshops hinausgeht. Insbesondere während des Entwickelns stofflich-digitaler Artefakte treten Probleme auf, bei deren Bewältigung andere Maker_innen Hilfe geben könnten, sofern das unfertige Projekt (bzw. dessen Modelle) anhand einer Dokumentation nachvollziehbar ist. Auch sind Projektdokumentationen Voraussetzung, um einmal angefangene Projekte zu einem späteren Zeitpunkt weiterzuführen oder an andere weiterzugeben, die die darin verfolgten Ansätze aufgreifen. Entsprechend müssen solche Projekte – einschließlich ihrer formalen informatischen Aspekte bzw. Modelle – verständlich abgebildet werden. Bislang ist wenig bekannt über geeignete Tools und Modellierungsmethoden, die verständlich und anwendbar für noch unerfahrene junge Maker_innen sind; Modellierungstechniken und -sprachen aus professionellen Bereichen sind Anfänger_innen nicht unbedingt zugänglich. Daher will diese Arbeit untersuchen, wie junge Menschen ohne Fachwissen Projekte stofflich-digitaler Artefakte nachvollziehbar darstellen können.

Die vorliegende Arbeit situiert das spielerische Konstruieren mit Construction Kits für Physical Computing, wie es in TechKreativ-Workshops angeboten wird, im Kontext der Makerbewegung. Sie widmet sich den *Modellen*, die dabei entstehen und zeigt Wege auf, wie junge ‚Selbermacher_innen' beim Modellieren und Dokumentieren der Modelle stofflich-digitaler Artefakte unterstützt werden können. Durch ihre Ergebnisse lenkt sie – soviel sei hier vorweggenommen – den Fokus auf Potenziale Visueller Programmierumgebungen als Dokumentationstools für Amateur_innen. Die Arbeit ist einzuordnen in die Bereiche IDC (Interaction Design and Children) und Informatik im Gebiet der Human-Computer Interaction. Sie leistet einen Beitrag zur Gestaltung von Dokumentationstools für junge Maker_innen, liefert Erkenntnisse zu Modellen in Makerkontexten anhand empirischer Untersuchungen und entwickelt einen modelltheoretischen Ansatz, um ‚Making' als Modellbildung zu beschreiben.

Ein Ziel dieser Arbeit ist es, das aus der Informatik entlehnte Modellkonzept auf das Selbermachen stofflich-digitaler Artefakte[3] im Kontext Makerbewegung zu übertragen. Es wird angenommen, dass eine Übertragung der Modellbildungskonzepte auf ‚Making' dazu beitragen kann, Ansatzpunkte und Konzepte für die Entwicklung geeigneter Modellierungs- und Dokumentationswerkzeuge für junge Amateur_innen zu finden. Dabei interessiert, was selbstgemachte stofflich-digitale Artefakte und deren Entstehungsprozesse kennzeichnet und welche Abbildungsmöglichkeiten sich daraus ableiten lassen, die auch für Menschen mit wenig Erfahrung und ohne technisches oder informatisches Fachwissen anwendbar und verständlich sind.

Auch hat diese Arbeit zum Ziel, Softwaretools entsprechend der theoretischen Konzepte zu implementieren und zu untersuchen. Die Modellierungstools sollen angehende Maker_innen dabei unterstützen, den Zustand – aktuelle Modelle – ihrer Projekte

[3] In der englischsprachigen Fachliteratur finden sich Begriffe wie ‚digitally enhanced artefacts' oder ‚digital physical artefact'. Hier soll der Begriff *stofflich-digitales Artefakt* verwendet werden. Damit soll betont werden, dass ein wesentlicher Bestandteil des Konstruieren eines solchen Artefakts das Erstellen des individuellen stofflichen Körpers ist (siehe auch Abschnitt 2.1.2).

und ihr Vorgehen verständlich und nachvollziehbar abzubilden. Diese könnten letztendlich in der Praxis dienlich sein, um in der Freizeit verfolgte Projekte zu späteren Zeitpunkten wieder aufzugreifen, an andere weiterzugeben oder Teile davon für andere Projekte weiterzuverwerten und so an der Makerbewegung zu partizipieren. Kollaborative Aspekte wie der Austausch von Dokumentationen sind jedoch nicht Gegenstand der Untersuchungen.

Folgenden Fragen geht diese Arbeit nach:

- Wie kann das Konzept der Modellbildung informatischer Systeme auf das Selbermachen stofflich-digitaler Artefakte übertragen werden?
- Wie können junge Amateur_innen unterstützt werden, stofflich-digitale Artefakte während des Konstruktionsprozesses umfassend und nachvollziehbar als Modelle abzubilden? Was für Tools eignen sich und welche Gestaltungsempfehlungen lassen sich daraus ableiten?

Diese Fragen werden später (insbesondere in Abschnitt 7.1) ergänzt durch untergeordnete Fragen, die sich auf einzelne, sich in der Arbeit entwickelnde Aspekte beziehen. Beispielsweise, was die von jungen Amateur_innen dokumentierten Modelle kennzeichnet und wie mit formalen Eigenschaften umgegangen wird oder welche Eigenschaften eines Artefakts als relevant gelten können.

Diese Arbeit geht zunächst theoriegeleitet vor. Sie nähert sich dem Modellieren stofflich-digitaler Artefakte im Makerkontext theoretisch über selbstgemachte Artefakte und deren Entstehungsprozesse. Dazu wird mit der „Allgemeinen Modelltheorie“ (AMT) von Stachowiak (1973) ein Konzept genutzt, anhand dessen Modelle nicht nur in der Informatik, sondern auch auf anderen Gebieten identifiziert werden können. So wird das Erstellen stofflich-digitaler Artefakte als Modellierung nicht nur in der Informatik, sondern auch im Design betrachtet, um alternative Modellierungskonzepte heranzuziehen und auf die Tätigkeiten des Selbermachens stofflich-digitaler Artefakte und beteiligte Modelle zu übertragen. Auch liefert die AMT ein Kategoriensystem, um Modelle anhand ihrer strukturellen Eigenschaften einzuordnen. Diese Kategorien werden genutzt, um Modelle, die junge Amateur_innen mit Modellierungstools sowie mit analogen Methoden erstellt haben, mittels einer Inhaltsanalyse nach Mayring (2010) zu untersuchen.

In die Praxis übertragen werden die theoretischen Erkenntnisse anhand von Softwareentwicklungen, die später in Anlehnung an design-based research iterativ weiterentwickelt werden. Modellierungsprozesse Jugendlicher mit diesen Tools werden in einer Fallstudie evaluiert, um die Situationsbezogenheit zu berücksichtigen. Dabei erhobene Daten werden qualitativ mittels Inhaltsanalyse (siehe oben) ausgewertet. Sie tragen dazu bei, auch die Ergebnisse der Modellanalyse zu kontextualisieren.

Im Anschluss an die Einleitung führt Kapitel 2 in den Kontext der Arbeit – die so genannte Makerbewegung – ein. Dieses Phänomen wird durch die Rolle, die Digitale

Medien dabei als Kommunikationsmedium, Werkzeug und Zielobjekt einnehmen, charakterisiert. Es werden gesellschaftliche Aspekte des Selbermachens herangeführt und betrachtet, wie und warum Selbermachende ihre Projekte mit ‚How-tos' dokumentieren und wie diese zum eigenständigen Aneignen neuer Konzepte beitragen können. Dazu werden handlungsorientierte Lerntheorien wie der Konstruktionismus mit ‚Making' in Verbindung gebracht.

Als theoretische Grundlage für das Konzept des Entwickelns informatischer Systeme als Modellbildung wird in Kapitel 3 die Allgemeine Modelltheorie (Stachowiak, 1973) eingeführt, die Modelle sehr generell definiert als reduzierte Abbilder von Originalen, die zu einem bestimmten Zweck von Subjekten erstellt werden. Davon ausgehend werden verschiedene Aspekte der – meist konzeptuellen – Modellbildung in der Informatik betrachtet. Auch werden ihre Grenzen und Herausforderungen aufgezeigt.

Anschließend wird in Kapitel 4 die zuvor eingenommene Sicht des Erstellens informatischer Systeme als Modellbildung auf das interdisziplinäre Gebiet der Human-Computer Interaction übertragen, weil dieses alternative Konzepte zu konzeptueller Modellierung liefern kann. Das Modellierungskonzept wird um Auffassungen und Praktiken aus dem Design, insbesondere dem Interaktionsdesign, erweitert. Dies eröffnet Herangehensweisen an das Entwerfen stofflich-digitaler Artefakte, die sich durch iteratives, reflektierendes Vorgehen anhand verschiedenster konkreter Modelle kennzeichnen. Ergänzt wird diese Perspektive durch Modellbetrachtungen im Mechatronikbereich, die aufzeigen, wie sich konkrete Herangehensweisen mit abstrakter Modellierung verbinden lassen.

In Kapitel 5 werden Modellierungsmethoden und -tools betrachtet, die für Amateur_innen geeignet erscheinen und/oder unformale Zugänge bieten, u.a. Storyboarding, Dokumentationstools und Programmierumgebungen für Maker_innen. Sie zeigen unformale Modellierungsmöglichkeiten auf oder nutzen konkrete Herangehensweisen an formale Modellierung. Auch werden Gestaltungsempfehlungen für solche Tools herangezogen.

Im sechsten Kapitel werden die zuvor getätigten theoretischen Betrachtungen miteinander in Verbindung gebracht und auf das Modellieren stofflich-digitaler Artefakte – Making – übertragen. Es werden wesentliche Aspekte selbstgemachter stofflich-digitaler Artefakte identifiziert und als Modellperspektiven formuliert. Von diesen Überlegungen ausgehend werden Anforderungen an Modellierungstools formuliert. Weil diese sehr vielfältig sind, wird als Konsequenz sowohl eine Visuelle Programmierumgebung erweitert als auch ein neues webbasiertes Modellierungstool gemäß der Erkenntnisse konzipiert und implementiert.

In Kapitel 7 werden – nach einem Überblick über die gewählte Methodik – Modelle, die junge Amateur_innen mit verschiedenen Tools und Techniken erstellt haben, hinsichtlich ihrer dargestellten Perspektiven auf das zu modellierende Artefakt inhaltlich analysiert. Die Analyse wird ergänzt durch eine Fallstudie, die die Eignung

der in Kapitel 6 entwickelten Modellierungstools in der Praxis evaluiert. Die Evaluationsergebnisse verweisen auf die Eignung ikonischer Bildmodelle zum Abbilden stofflich-digitaler Artefakte und zeigen Potenziale Visueller Programmiersprachen zur Dokumentation von Projekten auf. Anhand der Ergebnisse werden die zuvor erstellten Anforderungen revidiert und Empfehlungen formuliert, die in ein neues Modellierungstool einfließen, das eine Visuelle Programmierumgebung um zusätzliche Modellierungsfunktionen erweitert.

Im letzten Kapitel erfolgen ein Rückblick und eine Diskussion der Ergebnisse der Arbeit. Ihr Beitrag wird dargestellt und eingeordnet. Ausblickend werden Anknüpfungspunkte für weitere Forschung aufgezeigt.

Kapitel 2

Digitale Medien selbstgemacht im Kontext Makerbewegung

Seit einigen Jahren ist Selbermachen wieder ‚in'. Dieser Trend wird gar enthusiastisch als „DIY culture" (Paulos, 2012, S. 57) oder „maker culture" (Sharples et al., 2013, S. 33) gepriesen. Nicht nur statische, handwerkliche Objekte machen Menschen ‚selber'. Auch entwerfen, programmieren und konstruieren sie technische Artefakte. Insbesondere diese Gruppe wird als *maker* bezeichnet, benannt nach der Zeitschrift „Make"[1] (Mota, 2011, S. 283). Die dort präsenten ‚Macher_innen' betätigen sich nicht nur handwerklich, sondern auch als Amateur-Interaktionsdesigner_innen und -Informatiker_innen, indem sie mittels Open-Source-Hardware „technologically enhanced arts and crafts" (Mota, 2011, S. 283) erstellen. Sie haben dabei oft keine entsprechende professionelle Ausbildung, sondern erfinden und entwickeln auf Basis informell erworbener Kenntnisse informatische Systeme. Ihr Vorgehen bildet sich in zahlreichen, von ihnen erstellten Anleitungen und Dokumentationen ihrer Projekte ab, die im Internet zu finden sind. Verbunden ist damit eine bestimmte Haltung, auf die in diesem Kapitel unter anderem eingegangen wird. Insbesondere interessiert in dieser Arbeit, wie Digitale Medien diese DIY-Bewegung kennzeichnen. Was motiviert Selbermachende, und hat DIY gesellschaftliche Relevanz? Ferner soll in diesem Kapitel betrachtet werden, wie sich Maker_innen neue Fertigkeiten aneignen, wie sie ihre Projekte kommunizieren und wie sich dies im Internet abbildet.

2.1 Digitale Medien und Making

Schon immer haben Menschen gerne Dinge selbst gemacht. Auch so genannte DIY-Kulturen sind kein neues Phänomen. Als Beispiele werden in der Literatur Subkulturen genannt, die sich im Kontext von Punk-, Hippie- oder Hacker-Bewegungen im

[1] `http://makezine.com` (aufgerufen am 06.01.2013).

vergangenen Jahrhundert vornehmlich in den USA und in Großbritannien bildeten (Kuznetsov & Paulos, 2010; Anderson, 2013). ‚Selbermachen' schloss dabei nicht nur das Erstellen handwerklicher Artefakte, sondern auch das Produzieren und Publizieren von Zeitschriften, Radiosendungen und Musik ein. Dahinter steckten häufig gesellschaftliche und politische Unzufriedenheit mit einer Konsumgesellschaft und der Wunsch nach Individualität oder Nachhaltigkeit (Anderson, 2013).

Eine DIY-Bewegung ist also nichts Neues. Aber durch technologische Entwicklungen wie dem Web2.0 wird diese Makerbewegung sichtbarer als andere DIY-Bewegungen und wird teilhabbar unabhängig vom aktuellen Ort. Chris Anderson bezeichnet Maker_innen als Teil einer „Webgeneration" (Anderson, 2013, S. 33), für die es normal geworden ist, ihre Gedanken und Aktivitäten mit anderen online zu teilen.

Die Makerbewegung wird mit DIY-Aktivitäten von traditioneller Handarbeit und Basteln bis hin zu Ingenieurkonstruktionen und Amateurwissenschaft in Verbindung gebracht. Was diese Selbermachenden als Maker_innen auszeichnet und wie sie hier definiert werden sollen, ist die Tatsache, dass Digitale Medien Dreh- und Angelpunkt ihrer kreativen Tätigkeit sind. Dabei sollen drei Ebenen unterschieden werden, wie Digitale Medien in solchen Projekten involviert sind.

In den meisten Makerprojekten dienen Digitale Medien als Web2.0-Plattformen der Kommunikation, dem Austausch von Ideen, der Recherche und/oder der Präsentation fertiger Projekte mit dem Anliegen, Feedback zu erhalten. Das Teilen von Informationen und Projektdokumentationen in Online-Communitys spielt dabei eine wesentliche Rolle. Der deutsche Begriff des *Heim*werkens, der das Verortetsein von Selbermachen im privaten, häuslichen Kontext widerspiegelt, ist damit zur Beschreibung der Makerbewegung zu beschränkt. Denn Heimwerken wird nun ergänzt und bereichert durch *Netz*werken im öffentlichen Raum. Der Austausch mit Gleichgesinnten beschränkt sich nicht mehr auf die lokale Umgebung. Das selbstgemachte Objekt – bzw. eine Dokumentation dessen – verlässt die Garage oder den Hobbyraum und wird für andere Menschen weltweit sichtbar. Die Selbermachenden bekommen Anerkennung und Rückmeldung über ihr unmittelbares persönliches Umfeld hinaus, können aber auch Projekte Gleichgesinnter aufgreifen und weiterentwickeln. Das Darstellen der eigenen Projekte erfordert aber eine Dokumentation, die die wesentlichen Merkmale des Objekts nachvollziehbar für andere darstellt. Eine Reduktion oder Abstraktion auf die wesentlichen, für die Rezipient_innen relevanten Merkmale ist erforderlich. Zur Verfügung stehen dazu in der Regel Bild, Text und Video. Auch wird das Objekt durch die Dokumentation – zumindest teilweise – dekontextualisiert. Indem Heimwerken durch Netzwerken bereichert wird, ändern sich wiederum dessen Anforderungen (bzw. Herausforderungen) und Möglichkeiten.

Digitale Medien dienen in DIY-Projekten auch als (Modellier-)Werkzeug, um Artefakte zu erstellen. Am Computer werden digitale Modelle konstruiert, gezeichnet oder programmiert und anschließend von Fabrikationsmaschinen materialisiert. Ein

Beispiel sind digitale dreidimensionale Modelle, die mit 3D-Druckern ausgedruckt werden. Maschinen wie 3D-Drucker, Lasercutter, Vinylcutter oder Fräsmaschinen, die anhand digitaler Vorlagen stoffliche Objekte erstellen, sind mittlerweile über FabLabs auch Amateur_innen zugänglich (siehe Abschnitt 2.4.1).

Vielfach sind Digitale Medien nicht nur (Modellier-)Werkzeug, sondern als stofflich-digitale Artefakte auch eigentliches Ziel von Makerprojekten. Die Artefakte werden materiell konstruiert und enthalten Mikrocontroller, die programmiert werden müssen.[2] Auf diese dritte Stufe von Makerprojekten – dem Selbermachen programmierbarer stofflich-digitaler Artefakte – konzentriert sich diese Arbeit. In ihrem weiteren Verlauf steht *Making* deshalb für das Selbermachen stofflich-digitaler Artefakte.[3]

2.1.1 Online Communitys

> „The growing interest in DIY is charging a virtuous circle—individuals who make things enjoy documenting their projects online, which inspires others to try making them, too.“ (Frauenfelder, 2011, S. 222)[4]

Wie eingangs erwähnt, hat das Internet – insbesondere als Web2.0 – maßgeblich zum (Wieder-)Aufleben des DIY beigetragen. Das Internet bietet einen Reichtum an schnell zugänglichen Ressourcen und Hilfestellungen, die in einer Offline-Kultur i.d.R. nicht so einfach zugänglich sind. Es eröffnet Austauschmöglichkeiten für Menschen, die an einer speziellen Thematik interessiert sind und in ihrem lokalen Umfeld nur wenige Gleichgesinnte finden.

Um ein Projekt zu starten, brauche ich nur eine Suchmaschine nach meiner Idee zu befragen, schon finde ich Inspiration und Anleitungen, die meiner Idee entsprechen oder die mich in der Umsetzung weiterbringen. Sind mir die dazu notwendigen Techniken nicht vertraut – weiß ich z.B. nicht, wie man eine LED anlötet – so finde ich jede Menge Anleitungen, darunter eine Vielzahl von Amateur-Videos, die mir dies zeigen. Brauche ich ein spezielles Bauteil – z.B. leitfähigen Stoff (den der örtliche Kurzwarenladen sicher nicht vorrätig hat) – so finde ich zahlreiche, auch kleine Onlineshops, wo ich es mir sofort bestellen kann. Habe ich technische Probleme bei der Umsetzung meines Projekts, kann ich Foren durchstöbern oder selbst einen Thread mit meinem Problem eröffnen. So finde ich sicher schneller eine Lösung, als wenn ich tagelang das Problem

[2] Bei der Entwicklung solcher Physical-Computing-Projekte sind Digitale Medien auch in den zuvor beschriebenen Weisen involviert – es werden nicht nur Digitale Medien konstruiert und programmiert, sondern es werden z.B. digitale Tools benutzt, um diese zu programmieren, oder Teile dafür mit 3D-Drucker oder Fräsmaschine herzustellen. Und oft werden diese Projekte auch dokumentiert und so in Web-Communitys mit anderen geteilt.

[3] Um ein Gesamtbild des Phänomens Makerbewegung zu vermitteln, werden in diesem Kapitel auch Quellen herangezogen, die den Begriff des Making u.U. anders verwenden, als er hier definiert wurde.

[4] In seinem autobiografischen Buch „Made by Hand: My Adventures in the World of Do-It-Yourself“ (Frauenfelder, 2011), erzählt Mark Frauenfelder, Mitgründer und Chefredakteur des Magazins „Make“, von seinen eigenen DIY-Versuchen und Begegnungen mit Maker_innen. Auch wenn dies kein wissenschaftliches Werk ist, so gibt es interessante Einblicke in die Makerszene und wird hier exemplarisch herangezogen.

alleine versucht hätte zu lösen. Mein fertiges Projekt kann ich stolz mit Anleitung auf einem DIY-Internetportal veröffentlichen oder in meinem eigenen Blog präsentieren. Solche Möglichkeiten werden auf Webplattformen geboten, von denen einige weiter unten vorgestellt werden.

Auf der Plattform Makezine[5] der Zeitschrift Make werden vorwiegend Anleitungen für Technik-Projekte u.a. mit Arduino-Mikrocontroller-Boards vorgestellt. Die Projekte sind meist von männlichen Makern erstellt. Handwerkliche und technikorientierte Bastlerprojekte dominieren. Eine ähnliche Plattform ist Instructables[6]. Dort werden ebenfalls DIY-Projekte mit Anleitung vorgestellt, allerdings nicht nur auf Technik bezogene, sondern auch z.B. Kochrezepte. Während viele der Projekte auf Makezine von den Editor_innen des Magazins stammen oder von diesen begutachtet wurden, handelt es sich bei den Projekten auf Instructables hauptsächlich um User-generated Content.

Auf diesen Plattformen werden Projekte als so genannte *How-tos* eingestellt. Dabei handelt es sich um Schritt-für-Schritt-Anleitungen, die im Wesentlichen Bilder und Text enthalten. Dieses Prinzip wird weiter unten näher beschrieben und analysiert.

Für das Selbermachen von Dingen mit so genannten Fabbing-Technologien wie 3D-Druckern oder Lasercuttern existieren diverse Plattformen, auf denen Anwender_innen ihre digitalen 3D-Modelle tauschen, kollaborativ an diesen arbeiten und teilweise auch als 3D-Ausdruck bestellen können. Beispiele sind Thingiverse[7], Ponoko[8] oder Shapeways[9]. Weitere exemplarische Webseiten, auf denen ‚analoge' Projekte ausgetauscht werden, sind z.B. Ravelry[10] für Strickbegeisterte oder Ikeahackers[11] für eigene Modifikationen von in Massen produzierten Möbeln. Verkaufsplattformen wie Etsy[12], Cargoh[13] oder Dawanda[14] öffnen Vermarktungsmöglichkeiten für selbst hergestellte Produkte. Gleichzeitig werden all diese Plattformen genutzt, um Inspiration und Ideen für neue Projekte zu finden (Kuznetsov & Paulos, 2010). Crowdfunding-Plattformen wie Kickstarter[15] helfen dabei, Finanzierung für ambitioniertere Projekte durch Kleinstspenden zu sammeln.

Neben den genannten Plattformen gibt es auch zahlreiche DIY-Blogs, in denen vorwiegend Selbermacherinnen zu Crafting- und Dekorations-Themen ihre oft ästhetisch und künstlerisch ambitionierten Ideen, Projekte und Entdeckungen posten.

[5] `http://makezine.com` (aufgerufen am 06.01.2013).
[6] `http://www.instructables.com` (aufgerufen am 07.01.2013)
[7] `http://www.thingiverse.com` (aufgerufen am 07.01.2013)
[8] `https://www.ponoko.com` (aufgerufen am 07.01.2013)
[9] `http://www.shapeways.com` (aufgerufen am 07.01.2013)
[10] `http://www.ravelry.com` (aufgerufen am 07.01.2013)
[11] `http://www.ikeahackers.net` (aufgerufen am 07.01.2013)
[12] `http://www.etsy.com` (aufgerufen am 07.01.2013)
[13] `http://www.cargoh.com` (aufgerufen am 07.01.2013)
[14] `http://www.dawanda.de` (aufgerufen am 07.01.2013)
[15] `https://www.kickstarter.com` (aufgerufen am 07.01.2013)

2.1.2 Open-Source-Hardware

Die technologische Entwicklung Digitaler Medien in den vergangenen Jahren hat auch dazu geführt, dass die Konstruktion von Computersystemen günstiger und einfacher geworden ist und nicht mehr Informatiker_innen oder Ingenieur_innen vorbehalten bleibt. Verhältnismäßig mächtige informatische Systeme können heutzutage auch Amateur_innen zu Hause selbermachen. Dabei beschränken sich die Entwicklungsmöglichkeiten nicht nur auf Software. Dank Open-Source-Hardware wie z.B. Arduino[16] oder Raspberry Pi[17], die sich auch an Menschen ohne technische Berufsausbildung richten, ist es auch Laien möglich, kleine ‚ubiquitäre' Computersysteme zu konstruieren und zu programmieren.

Geschichte von Arduino

Arduino ist nicht nur ein Mikrocontroller-Board (siehe Abbildung 2.1), das Bestandteil vieler Makerprojekte ist, sondern auch ein Beispiel dafür, wie innerhalb dieser Makerbewegung durch Weitergabe von Projektdokumentationen Technologien weiterentwickelt werden.

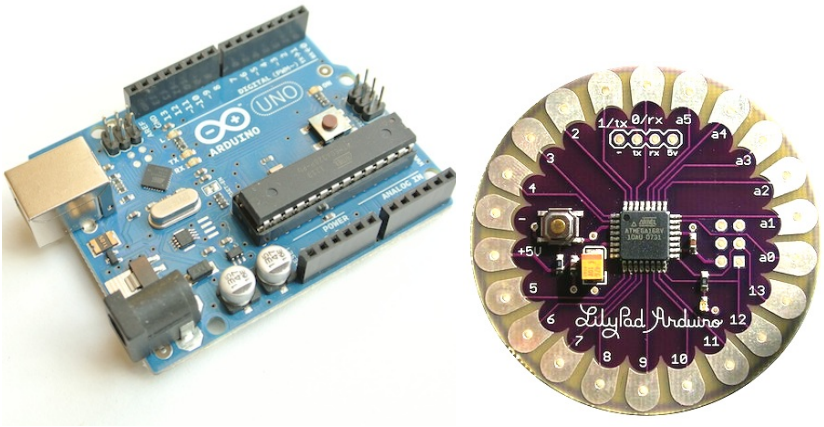


Abbildung 2.1: Arduino UNO und LilyPad Arduino Boards

Die Idee für das Arduino-Board entstand im Jahre 2005 an einer Designhochschule in Ivrea (Italien). Eine Gruppe aus Studierenden und Hochschuldozenten hatte die Idee, aufbauend auf einer Abschlussarbeit eines Studenten, ein eigenes Mikrocontroller-Board für den Unterricht in Interaktionsdesign zu entwickeln. Dieses sollte sich nicht, wie viele der bis dahin verfügbaren Boards, an technisch versierte Menschen richten, sondern eine Hardware im Sinne von *Processing*[18]werden. Processing ist eine Programmiersprache, die sich an Menschen ohne spezielle technische

[16] `http://www.arduino.cc/` (aufgerufen am 06.01.2013)
[17] `http://www.raspberrypi.org` (aufgerufen am 06.01.2013)
[18] Processing Website `http://www.processing.org` (aufgerufen am 01.09.2013)

Ausbildung richtet, wie Designer_innen und Künstler_innen, die damit relativ einfach Visualisierungen programmieren können. Mit Arduino sollten sie auch in die Lage versetzt werden, einfache stofflich-digitale Prototypen zu entwickeln. Da die Hochschule in Ivrea vor der Schließung stand, entschieden sie sich, die Pläne des Arduino-Board als ‚open source' zu veröffentlichen, ein Konzept, das bis dahin eher für Software gebraucht wurde. Damit darf jede_r das Arduino weiterentwickeln und diese Entwicklungen auch vertreiben, vorausgesetzt, er/sie legt ebenfalls die dafür verwendeten Pläne offen (Anderson, 2013; Hernandez, Calvo & Alejos, 2010).

Mittlerweile gibt es zahlreiche Weiterentwicklungen und Versionen des Arduino-Boards. Diese werden nicht nur von der ursprünglichen Zielgruppe genutzt, sondern z.B. auch von Wissenschaftler_innen für Prototyping-Zwecke oder zu Bildungszwecken im Schulunterricht. Arduino verfügt über eine große Online-Community, die die Entwicklungen vorantreibt. Arduino-Erweiterungen werden auch für neue Produkte benutzt wie z.B. Fluggeräte oder 3d-Drucker. Auch Handy-Bausätze auf Arduino-Basis gibt es bereits. Das Arduino-Board zeichnet sich insbesondere durch die Programmierbarkeit mittels einer relativ einfach erlernbaren Programmiersprache, die auf Processing basiert, aus. Das Konfigurieren komplizierter Schnittstellen entfällt für die Anwendenden, und Programme können unmittelbar ‚mit einem Klick' kompiliert und auf das Board geladen werden.[19]

Eine Weiterentwicklung ist das *Arduino LilyPad* (Buechley, Eisenberg, Catchen & Crockett, 2008). Dies wurde von Leah Buechley (Buechley et al., 2008) um 2007 entwickelt. Dabei handelt es sich um ein Arduino-Board, das keine Steckerbuchsen hat wie das herkömmliche Board, sondern ringsum mit Löchern versehen ist, die einen breiten leitfähigen Rand haben (siehe Abbildung 2.1). Außerdem ist das Board kleiner und flacher und rund statt eckig. Der Grund für dieses Design ist, dass das LilyPad für die Verbindung mit Smart Textiles gedacht ist. Statt also Sensoren und Aktuatoren mit Kabeln in Pinn-Buchsen zu stecken, werden diese mit Hilfe von leitfähigem Garn angenäht. Das LilyPad und die zugehörigen Aktuatoren und Sensoren erinnern sowohl in Größe als auch Aussehen an eine Brosche bzw. Knöpfe.

Geschlechterverteilung bei Arduino-Projekten

Nach eigenen Beobachtungen ist der überwiegende Teil der Urheber_innen von Howtos, die Technologie-Projekte auf der Plattform Instructables dokumentieren, männlich. Auch auf Maker Faires ergibt sich ein ähnliches Bild.

Über die Demographie derjenigen Maker_innen, die mit Arduino-Technologie eigene Digitale Medien erschaffen, kann eine Untersuchung von Buechley und Hill (2010) Aufschluss geben. Sie haben anhand von Verkaufsdaten, online veröffentlichten Projektdokumentationen und Umfragen untersucht, wie sich das LilyPad Arduino verbreitete, wie es angenommen und weiterentwickelt wurde zwischen Oktober 2007

[19] Arduino Blog `http://blog.arduino.cc` (aufgerufen am 29.07.2013)

und November 2009. Sie kommen dabei zu dem Schluss, dass das LilyPad die Entwicklung einer neuen, einmaligen „engineering community“ (Buechley & Hill, 2010, S. 200) ermöglicht hat. In ihrer Studie vergleichen sie die Demographie der LilyPad-Anwender_innen mit der der Arduino-Nutzenden. Frauen machen unter den Arduino-Kund_innen eine kleine Minderheit aus (9 % weiblichen und ca. 13% unbekannten Geschlechts), bei den LilyPads hingegen einen signifikant höheren Prozentsatz (35 % weiblichen und ca. 8% unbekannten Geschlechts) (Buechley & Hill, 2010, S. 202). Bei den Projektanalysen ergibt sich ein ähnliches Bild (Arduino: 86% m, 2% w, LilyPad: 25% m, 65% w) (Buechley & Hill, 2010, S. 202). Aus den Zahlen schließen Buechley und Hill (2010), dass LilyPad-Kundinnen eher Projekte erstellen, dokumentieren und teilen als männliche Kunden, bzw. Kundinnen dieses eher benutzen, wenn sie sich eines gekauft haben. Das LilyPad macht demzufolge solche Mikrocontroller-Technologien nicht nur für eine breitere Zielgruppe zugänglich, sondern hat auch zu einer neuen Community von Entwickler_innen und Designer_innen geführt, anstatt nur einer bereits existierenden ein neues Tool zur Verfügung zu stellen (Buechley & Hill, 2010).

Beim Selbermachen – dem Erstellen stofflich-digitaler Artefakte – mit Arduino-Technologien überwiegen also die Selbermacher gegenüber den Selbermacherinnen. Produkte wie das LilyPad Arduino erlauben es, an vielfältige Interessen anzuknüpfen. So können sie dazu beitragen, dass Menschen mit eher künstlerischem als technologischem Interesse dazu angeregt werden, persönlich-bedeutsame stofflich-digitale Artefakte zu schaffen und sich zu vernetzen.

Stofflich-digitale Artefakte mit Arduino

Stoffliche Objekte, die programmierbar sind und insbesondere solche, die Arduino verwenden, werden in dieser Arbeit als stofflich-digitale Artefakte benannt. Digital sind sie, sobald das eingebaute Arduino ein Programm ausführt, das dem Artefakt gewissermaßen ‚Leben‘ einhaucht, ihm ein Verhalten gibt und eine Interaktion mit der Umgebung ermöglicht. Stofflich sind die Projekte, da sie nicht nur aus einer Software bestehen. Ein wesentlicher Teil ist ihre materielle Hardwarekonfiguration, bestehend aus Board, Sensoren und Aktuatoren, die als Schnittstellen für Interaktion notwendig sind. Das Artefakt, das so entsteht und in das die Technik ggf. eingebettet ist, wird als ein materielles, gegenständliches Objekt[20] wahrgenommen.

2.2 Gesellschaftliche Bedeutung der Makerbewegung

Open-Source-Hardware-Entwicklungen und die sich damit eröffnenden Möglichkeiten des Rapid Prototyping kommen auch professionell Tätigen zugute. Gleichzeitig können damit auch Menschen außerhalb von Forschungsinstitutionen selbstständig Tangible User Interfaces erfinden und prototypisch entwickeln. Williams, Gibb und

[20] Das kann auch eine Rauminstallation oder eine bestimmte Umgebung sein.

Weekly (2012) beschreiben, wie das Gebiet des Interaction Research von der Open-Source-Hardware-Entwicklung profitiert. Sie betonen, dass dadurch Innovation demokratisiert worden sei, wozu auch die Verfügbarkeit von Treffpunkten oder Lernorten wie Hackerspaces und Ressourcen im Internet beitragen. Maker_innen und ihre Projekte sind so auch in den Fokus der Wissenschaft, insbesondere der HCI, gelangt – wie auch in dieser Arbeit.

So wie Williams et al. (2012) eine Demokratisierung von Innovation durch Partizipation von Maker_innen sehen, behauptet auch Dale Dougherty, Gründer der Zeitschrift Make im Film „We are Makers“ (Driskell & Bardwell, 2013):

> „If we can get more people tinkering and participating we'll have more creative solutions to more problems. (...) (...) as a maker, you get to create and make things, you get to shape and change the world around you.“ (Driskell & Bardwell, 2013)

Chris Anderson (2013), ehemaliger Chefredakteur der Zeitschrift Wired[21], bringt die Makerbewegung vorwiegend mit den neuen Möglichkeiten digitaler Fabrikation in Verbindung und prophezeit gar eine neue industrielle Revolution in der Fertigung. Dies begründet er unter anderem damit, dass das Internet Innovation und Produktion demokratisiert habe, als auch mit den neuen Fabrikationsmöglichkeiten.

> „Dank der Desktop-Fabber und durch den vereinfachten Zugang zu Produktionskapazitäten kann jeder mit einer Idee Geld mit der Herstellung materieller Waren verdienen. Und zweitens kann auch jeder, dank des Internets, seine Waren weltweit verkaufen. Die Einstiegshürden für Unternehmer in die Produktion materieller Waren sinken rasant.“ (Anderson, 2013, S. 223)

Durch individualisierte Produkte und eine flexiblere Produktion könnten Marktnischen besetzt werden. Die Produktentwicklung ließe sich mittels Maker-Communitys organisieren und mittels Crowdfunding finanzieren (Anderson, 2013).

Diese Vorstellung bezüglich einer Demokratisierung von Innovation und Produktion ignoriert meiner Meinung nach, dass, abgesehen von der persönlichen Motivation, die Einstiegshürden, sich in die Entwicklung technischer Innovationen einzubringen, für Laien doch verhältnismäßig hoch sind. Ob sich also eine gesellschaftliche Relevanz der Makerbewegung in dieser Hinsicht entwickeln wird, bleibt abzuwarten.

Der britische Soziologe und Medientheoretiker David Gauntlett (2011) geht der Frage nach, warum alltägliche Kreativität wichtig für die Gesellschaft sei.[22] Seiner Meinung nach setzen sich Selbermachende[23] durch ihre Tätigkeit stärker mit ihrer Umwelt auseinander und knüpfen Kontakt zu anderen. Er leitet aus der Beobachtung, dass handwerkliche Tätigkeit nicht mehr nur von Professionellen dominiert wird, sondern im Internet viele Amateur_innen ihre Ideen und Werke teilen, einen Wandel von einer „sit-back-and-be-told culture“ (Gauntlett, 2011, S. 245) zu einer „making-and-doing

[21] `http://www.wired.com` (aufgerufen am 07.01.2013)

[22] „*Why* is everyday creativity important?“ (Gauntlett, 2011, S. 19)

[23] Zu deren Tätigkeiten gehören hier auch das Erstellen digitaler Inhalte wie z.B. Youtube-Videos oder Blogeinträge.

culture“ (ebenda) ab, den er aus psychologischer, politischer, philosophischer und ökonomischer Perspektive betrachtet. Er schlussfolgert, dass Selbermachen und der damit verbundene Austausch von Ideen und Dingen, sowohl in materieller als auch sozialer Hinsicht verbindend wirken. Als einen wesentlichen Aspekt des DIY sieht Gauntlett (2011, S. 56) auch die psychologische Dimension, das Gefühl des Empowerment, selbst etwas erledigen zu können anstatt sich an andere wenden zu müssen sowie die persönliche Bedeutsamkeit, die dadurch entsteht. Er fordert für die Zukunft u.a. verbesserte Möglichkeiten für Kommunikation, Austausch und Zusammenarbeit, um engagierte Interessengemeinschaften zu fördern, deren Mitglieder sich gegenseitig helfen (vgl. Gauntlett, 2011, S. 234). Gauntlett (2011) sieht für das Selbermachen im Kontext der aktuellen DIY-Bewegung gesellschaftliche Relevanz aufgrund der damit einhergehenden gesellschaftlich verbindenden und befähigenden Elemente.

2.3 Maker_innen und ihre Motivation

Wie bereits beschrieben wurde, ist Selbermachen nicht neu. Doch wenngleich früher (und immer noch in vielen Gebieten der Erde) Selbermachen auch durch knappe Ressourcen oder finanzielle Not motiviert war, so ist der aktuelle DIY-Trend in der westlichen Welt vorwiegend durch einen gewissen ‚Lifestyle‘ getragen. Selbermachen im Makerkontext und die dabei entstehenden Produkte gelten als „cool“ (Bean & Rosner, 2014). Nicht nur das Selbermachen mit digitalen Tools oder das Entwickeln stofflich-digitaler Artefakte, sondern auch Handarbeit und Handwerk haben mit dem Internet einen neuen Auftrieb bekommen. Maker_innen präsentieren sich mit ihren Artefakten stolz in Online-Communitys und tauschen dort Anleitungen und Anregungen aus. Doch wer sind diese „makers“, „enthusiasts“, „hackers“, „modders“, „hobbyists“ (Torrey & McDonald, 2007), „amateurs“ (Frauenfelder, 2011; Paulos, 2012), wie sie in der Literatur benannt werden? Und was motiviert sie zum Selbermachen?

Mark Fauenfelder gibt in seinen Erzählungen Einblicke in die Beweggründe, die Menschen in seiner Umgebung zum Selbermachen antreiben. Er berichtet u.a. von Herausforderung, Lernen, Forscherdrang, einer persönlichen Beziehung zu den Objekten, etwas mit den Händen zu tun anstatt virtuell, dem Erleben von Taktilität und Haptik, Kreativität, Erfindergeist, Unabhängigkeit, Kontrollgefühl, Abschalten vom Alltag und Meditation (Frauenfelder, 2011). Kuznetsov und Paulos (2010) nennen als Motivation für DIY-Projekte u.a. den Wunsch nach Selbstausdruck und das Ausleben von Kreativität, das Lernen neuer Fertigkeiten, unkäufliche Dinge zu kreieren oder das Personalisieren von Gegenständen. Dabei ist auffällig, dass die Kreativschaffenden und ihre Produkte oft nicht dem klassischen Heimwerker- oder Handarbeits-Image entsprechen, sondern dass sie als junge, ‚hippe‘ Menschen (oft mit Bezug zu Design und Kunst) auftreten. Viele der im Internet präsentierten Produkte weisen ein modernes Design auf und sind nicht auf ihren Nutzen beschränkt.

Hauptmotive, zu DIY-Communitys beizutragen, sind u.a. Inspiration zu finden, neue Konzepte zu lernen, Gleichgesinnte zu treffen, Rückmeldung zu eigenen Projekten zu bekommen (Kuznetsov & Paulos, 2010). Kuznetsov und Paulos (2010) identifizieren nach einer Umfrage unter 2600 Mitgliedern von DIY-Communitys ein Wertesystem, dass Werte wie „open sharing, learning, and creativity“ (Kuznetsov & Paulos, 2010, S. 295) höher bewertet als „profit and social capital“ (ebenda).

Anderson (2013) nennt neben der Nutzung Digitaler Medien als Tools zum Selbermachen und dem Verwenden standardisierter Dateiformate als ein weiteres wesentliches Merkmal für die Makerbewegung deren gemeinsame Werte: „Eine kulturelle Norm, nach der diese Entwürfe in Online-Communitys miteinander geteilt werden und gemeinsam mit anderen daran gearbeitet wird“ (Anderson, 2013, S. 33).

Selbermachen im Kontext der Makerbewegung ist demzufolge mit einer bestimmten Haltung verbunden. Diese ist geprägt von Werten wie Individualität und Kreativität, aber auch von gemeinschaftlichen Interessen – ähnlich dem Open-Source-Gedanken.

2.4 Orte für Maker_innen

Neben der vielen virtuellen Orte im Internet entstehen weltweit lokale Treffpunkte, in denen Maker_innen ihre Projekte verfolgen und sich austauschen. Dazu zählen zum Beispiel FabLabs, Makerspaces oder Hackerspaces[24].

2.4.1 FabLabs

Das erste FabLab – Fabrication Lab – entstand 2002 am MIT durch Neil Gershenfeld (2007). Daraus entwickelte sich eine Bewegung mit einem weltweiten Netzwerk. FabLabs stellen in einer High-tech-Werkstatt Produktionsmaschinen zur Verfügung, die bislang für Einzelpersonen schwer zugänglich waren. Solche Maschinen sind z. B. 3D-Drucker[25], 3D-Scanner, Lasercutter oder Platinen-Prägemaschinen (Walter-Herrmann & Büching, 2013). FabLabs wollen offen für alle sein und sind insbesondere für Designer_innen und Hobbybastler_innen interessant, die dort rasch Prototypen erstellen und in kleiner Serie fertigen oder auch Ersatzteile drucken können.[26] Die Idee der FabLabs ist eng mit der Makerbewegung verknüpft und teilt deren Werte. FabLabs sind aber auch Orte, an denen Leute mit Arduino-Technologie Artefakte konstruieren (und z.T. durch maschinell gefertigte Teile ergänzen).

[24] `http://hackerspaces.org/` (aufgerufen am 07.01.2013)

[25] 3D-Drucker sind mittlerweile auch als Bausatz zum Eigenbau erhältlich. Einige wurden und werden von Open-Source-Communitys entwickelt (z.B. der Makerbot `http://creativecommons.org/tag/makerbot` (aufgerufen am 12.03.2013).

[26] Was auch zu neuen Herausforderungen für das Urheberrecht führt.

Viele FabLabs, Makerspaces und vergleichbare Orte bieten Workshops an, um interessierten Laien einen Einstieg in bestimmte Technologien zu geben[27].

2.4.2 TechKreativ

Unter dem Titel *TechKreativ*[28] finden seit über sieben Jahren an der Universität Bremen in der Arbeitsgruppe *dimeb* temporäre Workshops für vorwiegend junge Menschen statt. In den mehrtägigen Kreativ-Workshops entwerfen, basteln und programmieren die Teilnehmer_innen eigene, persönlich-bedeutsame Projekte (Beispiele in Abbildung 2.2). Sie werden dabei von Tutor_innen betreut, das Lernen hat eher informellen Charakter und folgt konstruktionistischen Prinzipien (siehe Abschnitt 2.5.3) (Dittert, Katterfeldt & Schelhowe, 2012). Diese Workshops kooperieren mittlerweile eng mit dem FabLab Bremen e.V.[29].

Die Workshopteilnehmenden sollen in dieser Arbeit nicht nur als bastelnde Kinder in einem Medienbildungskontext wahrgenommen werden, sondern als angehende, temporäre Maker_innen: Die Rahmenbedingungen der TechKreativ-Workshops wie informeller Zugang, persönlich bedeutsame Projekte, Kreativität und kein vorausgesetztes Fachwissen entsprechen den Prinzipen, die der so genannten maker culture zugeschrieben werden.[30] Bezogen auf die zu Anfang des Kapitels vorgenommene Charakterisierung von Making, sind die Tätigkeiten in TechKreativ-Workshops vorwiegend auf der dritten Stufe einzuordnen, indem Digitale Medien als stofflich-digitale Artefakte selbstgemacht werden. Ein Austausch in virtuellen Communitys findet jedoch nur in sehr begrenztem Umfang statt (siehe Abschnitt 5.1.2).[31]

Diese Arbeit, die im Kontext von TechKreativ entstanden ist, konzentriert sich als Zielgruppe demzufolge auf junge angehende Maker_innen im Alter von ca. 13 bis 18 Jahren, die an diesen Workshops teilnehmen und mit Arduino-Technologie stofflich-digitale Artefakte erstellen.

2.5 Lernen beim Making

> „What I learned from Mister Jalopy and other DIYers is that mistakes are not inevitable—they're a necessary part of learning and skill building. Mistakes are a sign that you're active and curious." (Frauenfelder, 2011, S. 23f.)

[27] z.B. Miss Baltazar's Laboratory in Wien (`http://www.mzbaltazarslaboratory.org`, aufgerufen am 07.01.2013)

[28] `http://www.techkreativ.de` (aufgerufen am 07.01.2013)

[29] `http://www.fablab-bremen.org` (aufgerufen am 04.04.2014)

[30] Wobei nicht auszuschließen ist, dass die Motivation zur erstmaligen Teilnahme am Workshop z.B. auch durch die Eltern beeinflusst wird.

[31] Neuerdings werden auch TechKreativ-Workshops zum Thema Fabbing angeboten, in denen mit digitalen Tools Artefakte erstellt werden – und die damit auf der zweiten Stufe einzuordnen sind. Diese Art von Workshops wurden in dieser Arbeit nicht berücksichtigt.

Abbildung 2.2: Von Kindern und Jugendlichen in TechKreativ-Workshops selbst erfundene und konstruierte stofflich-digitale Artefakte: Vier Zauberwesen, Handschuh zum Musikmachen (oben Mitte), diebstahlsichere Handtasche (oben rechts)

Lernen voneinander und Lernen durch Machen, eingenommen Fehlermachen, charakterisieren, wie Maker_innen ihre Projekte umsetzen, sich neue Dinge aneignen und ihre eigenen Erkenntnisse weitergeben, so dass andere davon lernen können. Lernen geschieht dabei eher informell. Fertige Projekte werden oft in Anleitungsform, so genannten *How-tos*, präsentiert, unterstützt von Videos, schematischen Zeichnungen, Dateianhängen, vor allem aber durch viele Fotos der verschiedenen Arbeitsschritte. Plattformen wie Instructables bieten Maker_innen nicht nur Präsentationsfläche, sondern auch die Möglichkeit, Feedback wie Anregungen und Anerkennung aus der Community zu erhalten.

Auch findet Austausch insbesondere bei technischen Problemen in Foren statt. Foreneinträge beschränken sich aber oft auf Text und ggf. Links zu extern abgelegten Bildern oder Dateianhängen, z.B. Programmcode. Video-Dienste wie YouTube[32] bieten eine Fülle an Videos, in denen Arbeitsschritte, die oft handwerklicher Art sind, vorgeführt und erläutert werden.

2.5.1 How-tos

Maker_innen präsentieren ihre Projekte auf entsprechenden Plattformen in Form von Anleitungen (engl. ‚How-to Pages' oder ‚How-tos'), in denen sie detailliert Schritt-für-Schritt beschreiben, wie ihre Projekte (nach-)gebaut werden können. Wie „computer and electronic hobbyists" (Torrey & McDonald, 2007, S. 391) dieses neue Genre benutzen, um technische Probleme zu lösen, ihr Wissen für andere zu dokumentieren und wie sie sich gegenseitig helfen und Feedback geben, haben Torrey und McDonald (2007) untersucht. Demnach zeichnen sich How-tos dadurch aus, dass sie prozedurale Information sequentiell beschreiben. Dabei erzählen sie zum Beispiel chronologisch die Erfahrungsgeschichte der Erstellenden mit ihrem Projekt oder beschreiben das Projekt nachvollziehbar wie ein Rezept mit einer Liste der benötigten Werkzeuge und Materialien mit knappen, zielgerichteten Anweisungen (Torrey & McDonald, 2007). Das Prinzip der How-tos ist dabei nicht neu und seit den Anfängen des Internets vorhanden. Allerdings sei es durch die Möglichkeiten und Flexibilität des Web2.0 insbesondere im Umgang mit multimedialen Inhalten einfacher geworden, solche Anleitungen zu erstellen (Torrey, McDonald, Schilit & Bly, 2007).

Die Projekte der befragten Hobbyisten folgen einem bestimmten Zyklus mit verschiedenen Stufen des Wissenserwerbs und -austausches. Er beginnt mit dem Umsetzen des Projekts, gefolgt vom Erzählen der Geschichte des Projekts im How-to, das wiederum zum Beitrag für die Community wird und von anderen anerkannt und genutzt wird:

> „As hobbyists pursue each goal in turn, they may be engaged in knowledge retrieval, knowledge creation, knowledge exchange, or some combination." (Torrey & McDonald, 2007, S. 398)

[32] `http://www.youtube.com` (aufgerufen am 01.04.2013)

Es werden verschiedene Gründe genannt, warum Hobbyisten ihre Projekte als How-tos dokumentieren, z.B. um eine Reaktion von anderen zu erhalten, selbst der Community etwas zurückzugeben, um in der Szene bekannter zu werden und sich ein Portfolio zu erarbeiten, aber auch von einer Tagebuchfunktion wird berichtet. How-tos haben meist eine narrative, sequentiell strukturierte Form, in der auch Rahmenbedingungen, Motivation und Anwendung des fertigen Produkts erzählt werden. Erzählstil, Detailliertheit und enthaltene Elemente variieren stark. Insgesamt wird das Format der How-tos als flexibel beschrieben.[33] Die Texte werden durch diverse Medien (Fotos, verschiedene Grafiken, Videos) ergänzt. Diese werden meist erst nach Abschluss des jeweiligen Projekts erstellt (Torrey & McDonald, 2007).[34]

Neben der Struktur der How-tos untersucht die Studie auch, über welche weiteren Wege sich die Hobbyisten Hilfe holen. Dabei wird (positives) Feedback auf How-tos als wichtige Möglichkeit, Kontakt zu Gleichgesinnten aufzunehmen, erachtet. Es wird Kontakt in Expertenforen gesucht oder direkt per Email, beispielsweise von Leser_innen der How-tos, die um Hilfe fragen. Diese Art des Austausch wird als positiv beschrieben, insbesondere wenn es den Kontaktsuchenden dadurch gelingt, ein ähnliches Projekt erfolgreich durchzuführen. Um am Wissensaustausch mit Gleichgesinnten teilzunehmen, sei es für die Hobbyisten essentiell Präsenz zu zeigen, indem sie ihre Projekte veröffentlichen. Teilweise erlangen sie Informationen auch über persönliche Treffen mit anderen Maker_innen (Torrey & McDonald, 2007).

How-tos können somit als ein wichtiges Medium zum Wissensaustausch und Wissenserwerb innerhalb von Maker-Communitys angesehen werden, sowohl für die Erstellenden selbst, als auch für Rezipient_innen.

2.5.2 Praxisorientiertes Lernen mit Projektdokumentationen

Wie oben beschrieben, zeichnet sich die Makerbewegung dadurch aus, dass Selbermachende sich vernetzen, manche von ihnen sogar Digitale Medien selbst herstellen und Selbermachen damit zum Making wird. Um sich neue Fertigkeiten anzueignen, können die Selbermachenden auf How-tos und Erklärvideos anderer Maker_innen zurückgreifen. In How-tos wird i.d.R. Schritt für Schritt die Umsetzung eines Projekts geschildert (Abbildung 2.3). Das heißt, anhand eines konkreten Beispiels wird auf Bildern vorgeführt und im Text erklärt, wie z.B. die Hardwareteile verbunden werden müssen, wie der Programmcode aussehen sollte und ggf. wie beides zusammenhängt, aber auch wie die statischen materiellen Komponenten erstellt und zusammengebaut wurden. Insbesondere, um die Hardwareverbindungen darzustellen, werden oft Fo-

[33] Die von Torrey und McDonald (2007) untersuchten How-tos wurden auf eigenen Websites publiziert, und nicht mittels vorgegebener Templates wie z.B. auf Instructables.

[34] Dies könnte sich seit 2007 mit Plattformen wie Instructables und anderen geändert haben, weil die Akteur_innen sich möglicherweise daran gewöhnt haben, ihre Projekte dort einzustellen. Die meisten Fotos wirken zumindest nicht so, als wären sie im Nachhinein gemacht worden – z.B., weil dies das Dekonstruieren fest verbauter Teile erfordert hätte.

tos oder bildhafte anschauliche Grafiken (z.B. erstellt mit Fritzing, siehe Kapteil 5.3.4) verwendet.[35] How-tos enthalten oft auch ‚best-practice' Tipps, in denen aufgrund der eigenen Erfahrungen mit dem vorliegenden Projekt Hinweise gegeben werden, welche Probleme aufgetaucht sind und wie sie gelöst wurden. Manche Arduino-How-tos enthalten auch Videos, die das Endresultat vorführen, indem sie das erstellte Produkt ‚in Aktion' zeigen.[36]

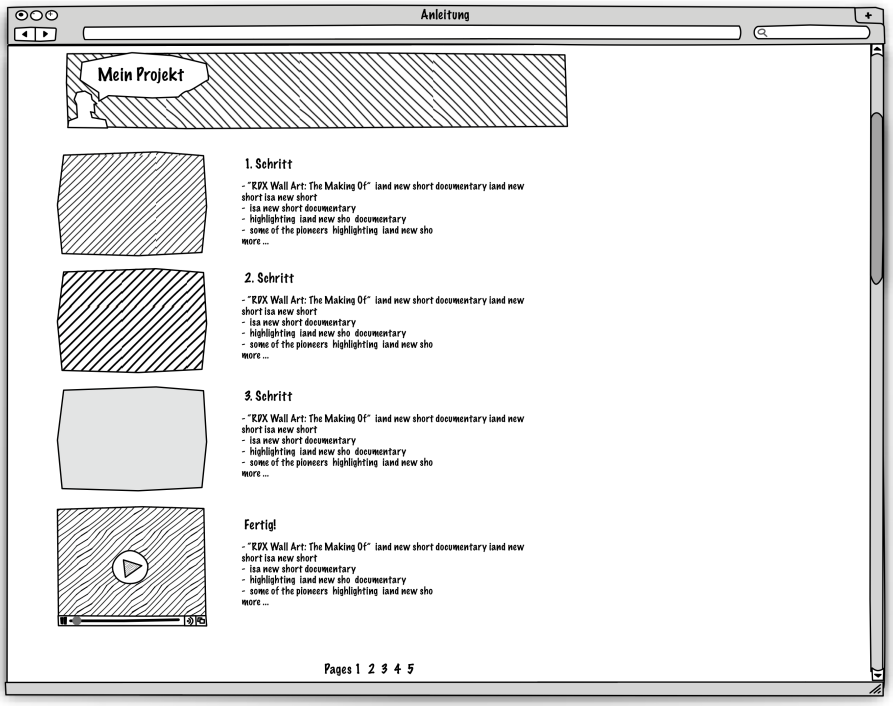


Abbildung 2.3: Typische Struktur eines How-tos

Erklärvideos sind oft kurz und konzentrieren sich i.d.R. auf bestimmte Handgriffe, die vermutlich für die Vorführenden intuitiv und schwer nur mit Worten zu erklären

[35] Ein exemplarisches Arduino-How-to: „Necomimi Arduino Cat Ears" `http://www.instructables.com/id/Necomimi-Arduino-Cat-Ears/` (aufgerufen am 03.02.2014). Weitere How-tos mit Arduino-Technologie: `http://www.instructables.com/tag/type-id/stepbystep-true/category-technology/channel-arduino/` (aufgerufen am 03.02.2014)

[36] Diese Aussagen beruhen auf einer stichprobenartigen Untersuchung von Arduino-Projekten auf der Plattform Instructables. Dazu wurden 19 How-tos mit Arduino-Projekten, die über den damaligen Sortierbutton ‚Random' ausgewählt wurden, analysiert. Ein Projekt enthielt durchschnittlich 6-7 Schritte, 14 Bilder und 740 Wörter beschreibenden Text (ohne Programmcode). Zwei Drittel der Projekte enthielten ein bis zwei Videos, von denen die meisten eine Demonstration des fertigen Produkts zeigten. Knapp die Hälfte hatte Dateianhänge (meist Skizzen oder Programmcode, wobei letzterer in vielen Projekten im Beschreibungstext enthalten war). Die meisten Autor_innen erwähnen die Motivation und Rahmenbedingungen ihrer Projekte. Über die Hälfte ließ als hauptsächliche Motivation erkennen, etwas für sich selbst oder Freunde/Familie zu erstellen, um diese zu erfreuen oder Lebensumstände zu verbessern. Fast eben so viele wollten einfach nur etwas erfinden oder ausprobieren. Die Analyse erfolgte im Zeitraum 5.1.-13.1.2012. Zu diesem Zeitpunkt befanden sich auf der Plattform ca. 300 Arduino-Projekte.

sind. Solche Videos finden sich vielfach, um z.B. Häkel-Maschen zu zeigen[37] oder im Technologie-Bereich, um vorzuführen, wie elektronische Komponenten an eine Platine gelötet werden.[38] Videos sind auch beliebt, um Installationsanleitungen zu geben, z.B. welche Schritte und Einstellungen am Bildschirm und am Arduino-Board notwendig sind, um ein Programm auf ein Arduino-Board zu laden und dort auszuführen.[39]

How-tos und Erklärvideos werden für deren Rezipient_innen zu Quellen informellen Lernens. Auf konkrete und allgemeinverständliche Weise werden anhand praktischer konkreter Beispiele Vorgehen und Sachverhalte dargelegt. Um z.B. eine LED an ein Arduino-Board anzuschließen und diese zum Leuchten zu bringen – ein typisches Anfängerprojekt –, finden sich zahlreiche Tutorials unterschiedlicher Detaillierungsgrade, die mit Fotos und auf die vorliegende Hardware bezogenen Erklärungen quasi am praktischen Beispiel ‚vormachen', wie eine LED angeschlossen wird, wie der Programmcode aussehen könnte, wie das Arduino-Board an den PC anzuschließen ist oder was sonst noch zu beachten ist.[40]

Um ein solches Problem wie das Anschließen und Programmieren einer LED zu lösen, müssen keine verallgemeinernden abstrakten Beschreibungen in Elektronik-Fachbüchern studiert werden oder die Referenz der Programmiersprache durchsucht werden.[41] Außerdem können verschiedene Medien einfacher eingebunden werden, die je nach Sachverhalt das erklärte zusätzlich veranschaulichen können, z.B. zeigen viele in How-tos eingebundene Videos die Interaktion mit dem programmierten Artefakt. Auch können hier digitale Projektteile, wie Programmcode oder ggf. Vorlagen für 3D-Drucker, über Dateianhänge direkt mitgegeben werden. Durch die Vielzahl an How-to-Autor_innen finden sich im Netz viele aktuelle Beispiele. Auf Rückfragen kann schnell reagiert werden, oft entwickeln sich unter der Kommentarfunktion der How-tos richtige Diskussionen.

Erstellende von How-tos sind offenbar meist erfahrene Maker_innen mit technischer Affinität, die ihr Wissen so weitergeben wollen und sich Rückmeldung und Anerkennung von anderen erhoffen. Für ein How-to müssen in den meisten Fällen während des Machens passende Fotos und Notizen erstellt werden. Bei der Erstellung des How-tos werden Handlungen, die sich als wesentlich erwiesen haben, ausgewählt und so beschrieben, dass sie nachvollziehbar sind, sich aber zugleich nicht in Details verlieren. Für How-tos oder Videos müssen deren Autor_innen den abzubildenden Pro-

[37] Z. B. Video „Häkeln lernen - ganzes Stäbchen" `http://youtu.be/LRQaGSJM2b8` (aufgerufen am 03.02.2014).

[38] Zum Beispiel das Video „How to Solder in 30 Seconds" `http://www.instructables.com/id/How-to-Solder-in-30-Seconds/` (aufgerufen am 03.02.2014)

[39] Zum Beispiel das Video „How to program an Arduino" `http://youtu.be/4HqXAmV_Ock` (aufgerufen am 03.02.2014).

[40] Siehe z.B. „Basic Arduino Tutorials : 01 Blinking LED" `http://www.instructables.com/id/Basic-Arduino-Tutorials-01-Blinking-LED/` (aufgerufen am 03.02.2014) für ein ausführliches Tutorial.

[41] Gewiss gibt es auch gute Bücher, die auf ähnliche praxisnahe Weise solche Anleitungen bieten. Allerdings sind Bücher – sofern nicht schon vorhanden oder als Ebook verfügbar – nicht so direkt und kostenlos zugänglich. Bücher über aktuelle Technologien sind auch oft schnell veraltet.

zess in seine wesentlichen Bestandteile gliedern. Auch übersichtliche Fotos, auf denen das Wesentliche zu Transportierende erkennbar ist, oder ruhige Kameraführung sind für Rezipient_innen relevant.[42] Das Erstellen solcher Projektdokumentationen kann als ein deutliches Zurücktreten aus dem Konstruktionsprozess betrachtet werden, das das Einnehmen der Rezipient_innenperspektive erfordert und demnach für die Erstellenden lernförderlich sein kann (siehe Abschnitt 2.5.3).

Den Rezipient_innen ermöglichen How-tos und Erklärvideos informelles Lernen, indem sie anhand der konkreten und i.d.R. allgemeinverständlichen Anleitungen durch aktives Konstruieren und Nachmachen eines vergleichbaren Projekts die Konzepte, die diesem innewohnen, erfahren können. Durch die Praxis-Orientierung der Anleitungen und weil die grundlegenden innewohnenden (ggf. abstrakten) Konzepte nicht ausdrücklich erläutert werden, kann jedoch nicht behauptet werden, dass How-tos oder Erklärvideos immer dazu führen, dass den Projekten und der Technologie innewohnende Konzepte erlernt werden. Es kann höchstens gemutmaßt werden, dass durch die Reihe an Erfahrungen, die so gesammelt werden, ein Verständnis dafür entsteht. Lernen bedeutet in diesem Kontext, sich diejenigen Fertigkeiten anzueignen, die notwendig sind, um das Projektziel zu erreichen und Spaß und Erfüllung beim Selbermachen zu erleben.

2.5.3 Bezug zu Lerntheorien

Informelles Lernen ist ein wesentlicher Bestandteil des Selbermachens. So können unter Heranziehen der beschriebenen Lernressourcen technische und informatische Projekte von Menschen erstellt werden, ohne dass diese sich mit den theoretisch-formalen Grundlagen im Detail beschäftigen müssen. Stattdessen können sie sich die notwendigen Fertigkeiten bei anderen mit Hilfe der beschriebenen bild-basierten Medien ‚abschauen' und durch eigenes Konstruieren aneignen. Hier sollen drei Ansätze vorgestellt werden, die mit dieser Form des Lernens theoretisch in Verbindung gebracht werden können.

Selbermachende lernen im Handeln. Der Philosoph und Pädagoge John Dewey (1859-1952), Vertreter des amerikanischen Pragmatismus, entwickelte einen handlungs- und erfahrungsorientierten Ansatz, wonach der Erwerb von Wissen eher durch Tun als durch Sehen erfolgt, was oft unter *Learning by doing* gefasst wird. Lernen besteht nach Dewey (1966) aus dem Sammeln von Erfahrungen. Zu einer Erfahrung (experience) gehört immer eine aktive Komponente, das aktive Handeln, und eine passive Komponente, die Konsequenz des Handelns und dessen Reflexion. Idealerweise sollen Lernende als aktive Wesen in einer kontrollierten Lernumgebung und einem Projekt – bevorzugt gemeinsam – durch experimentelles Handeln die Welt entdecken. Die Lernenden

[42] So wird zu Videos und How-tos oft kritisches Feedback gegeben, wenn die Erklärungen nicht ausreichend sind und z.B. für das Verständnis wesentliche Schritte fehlen.

entwickeln ihre Erkenntnisse und ihr Wissen aus den durch Handeln und Reflexion gemachten Erfahrungen (Dewey, 1966).[43]

Auch beim Making steht das Handeln im Vordergrund. Maker_innen erlernen Fähigkeiten, die sie für ihre Projekte benötigen, nicht nur durch eigenes Erproben und Erfahren, sondern relativ einfach auch durch Austausch mit anderen, z.B. Kommunikation über How-tos und Videos, aber auch durch Diskussionen in Foren und Sozialen Netzwerken. Dabei lernen sie Fertigkeiten von erfahreneren Maker_innen. Das Lernen in Interaktion mit anderen Selbermachenden, die bereits mehr Erfahrung gesammelt haben und anderen weiterhelfen können, beschreibt Lev Vygotsky (1978). Seine Lerntheorie ist im Konstruktivismus zu verorten. Er betont die Rolle von *Peers* (dt. etwa Fachkolleg_innen), die in ihrer kognitiven Entwicklung, d.h. ihrem Wissensstand, bereits weiter sind. Dazu unterscheidet Vygotsky (1978, S. 85) zwischen dem, was Lernende (z.B. Kinder) tatsächlich schon können aufgrund ihres Entwicklungsstands, und dem, was sie mit Hilfe eines Peers in der Lage wären zu tun. Diesen Bereich nennt er „Zone of Proximal Development“ (Vygotsky, 1978, S. 86):

> „It is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers.“ (Vygotsky, 1978, S. 86)

Lernziele sollten innerhalb dieser Zone liegen. Die Ansätze dafür sind schon vorhanden, durch Anleitung können Lernende sie entfalten und sich die jeweiligen Kenntnisse aneignen.

Beim Making findet die Kommunikation mit Peers jedoch über die oben beschriebenen Medien und damit asynchron statt. Dennoch ist das Lernen von Peers ein Charakteristikum, dass das Aneignen neuer Fertigkeiten im Kontext der Makerbewegung kennzeichnet. Die meisten How-tos und Videos werden nicht von Lehrer_innen oder anderen dafür ausgebildeten Fachpersonen erstellt, sondern von gleichgesinnten Maker_innen.

Eine Lerntheorie, die sich aus dem Konstruktivismus entwickelte, wird besonders häufig mit der Makerbewegung assoziiert: Seymour Paperts Constructionism (dt. Konstruktionismus) (Papert, 1980). So nennen Martinez und Stager (2013) Papert gar den „father of the maker movement“ (Martinez & Stager, 2013, S. 17) und formulieren enthusiastisch:

> „The maker movement is terribly exciting in the ways it celebrates the virtues of constructionism, even if the advocates of learning by making have no formal knowledge of the theory underlying their passions.“ (Martinez & Stager, 2013, S. 32)

Lernen passiert im Konstruktionismus durch aktives Konstruieren eines persönlich bedeutsamen Objekts. Dieses „object-to-think-with“ (Papert, 1980, S. 11) dient dazu,

[43] Später wird in Abschnitt 4.3.1 das Konzept von „reflection-in-action“ von Schön (1983, S. 49) beschrieben. Dieses Konzept baut auf den Lerntheorien von Dewey auf.

die eigenen mentalen Konzepte und darin (noch) enthaltene Fehler wahrnehmbar zu machen, zu erkennen und entsprechend zu revidieren. Abstrakte Prinzipien wie Programmierung und Elektronik können durch die Konstruktion des konkreten Objekts erlernt werden. Die Entwicklungspsychologin Edith Ackermann betont in Anknüpfung an Paperts Constructionism die Wichtigkeit von Reflexionsphasen für erfolgreiches Lernen neben der Objektkonstruktion. Neben dem Vertieftsein braucht es auch immer wieder Momente, in denen die Lernenden zurücktreten und mit Distanz das Geschehene betrachten. So entsteht ein Wechsel zwischen „diving-in“ und „stepping-out“ (Ackermann, 1996). Die Lernenden nehmen die Perspektive von Außenstehenden an, um sich anschließend wieder vertieft mit dem Lerngegenstand zu befassen.

So-genannte *Construction Kits*, in denen diese Lerntheorie einen Ausdruck findet, stellen Materialien bereit, denen ein abstraktes Konzept zugrunde liegt, das durch den Konstruktionsprozess konkret erfahren werden kann. Die Lernenden nehmen dazu die Rolle von Entwerfenden ein. Beobachtet man beispielsweise die Entwicklung Visueller Programmierumgebungen oder Construction Kits für Physical Computing, so fällt auf, dass die erhältlichen Tools sich vor wenigen Jahren noch hauptsächlich als ‚konstruktionistische‘ Lernmedien an Kinder und Jugendliche richteten, wie z.B. Lego Mindstorms[44] oder Cricket[45] mit ihren entsprechenden Visuellen Programmierumgebungen. Mittlerweile richten sich sowohl Softwaretools (siehe Kapitel 5) als auch Technik-Baukästen[46], die als Nachfolger der oben genannten Kits betrachtet werden können, auch an Erwachsene bzw. werden von diesen offenbar genutzt.

Die Makerbewegung macht sowohl digitale (z.B. Anleitungen) als auch stoffliche (z.B. Arduino-Kits) Ressourcen zugänglich, die ein eigenständige Aneignen informationstechnischer Sachverhalte durch produktives, erfindendes kreatives Handeln ermöglichen. Damit können sich auch Laien zu Hause mit neuartigen Technologien beschäftigen, indem sie stofflich-digitale Artefakte konstruieren. Dabei können sie etwa über informatische Prinzipien erfahren und lernen. Sofern diese Tätigkeit Einblicke in die ihnen zugrunde liegenden abstrakten Konzepte geben, kann behauptet werden, dass die mit der Makerbewegung assoziierten Ressourcen konstruktionistisches Lernen für eine breite Masse eröffnen.

Doch wie eignen sich diese Ressourcen – insbesondere How-tos – für das Lernen über Digitale Medien anhand der Konstruktion stofflich-digitaler Artefakte?

2.5.4 Lernen über Digitale Medien

Stofflich-digitale Artefakte – Computer – beruhen auf formalen mathematischen algorithmischen Anweisungen. Diese Abstraktionen sind über das Interface und die Inter-

[44] `http://mindstorms.lego.com` (aufgerufen am 24.02.2013)

[45] `http://www.handyboard.com/cricket/` (aufgerufen am 24.02.2013)

[46] z.B. die Fritzing Kits `http://shop.fritzing.org/products/fritzing-starter-kit-with-arduino-uno` (aufgerufen am 12.09.2013)

aktion i.d.R. nicht unmittelbar wahrnehmbar. Interagierende erleben das Verhalten des Artefakts mit ihren Sinnen auf unformale, konkrete Weise.

Der stoffliche Teil des Artefakts als sein statischer Körper kann zu einem bestimmten Zeitpunkt als vollendet angesehen werden, sobald es für die Schaffenden seinen Zweck erfüllt. Das interaktive programmierte Verhalten hingegen bleibt unfertig und somit das stofflich-digitale Artefakt. Denn erst in der Interaktion mit dem (nachgebauten) Artefakt, die unendlich viele Formen annehmen kann, entsteht es. Dabei ist es nie vollendet durch die unendlichen Möglichkeiten der Interaktion (Schelhowe, 2007; Lunenfeld, 2000). Typische How-tos, die stofflich-digitale Artefakte dokumentieren, enthalten Bilder und Beschreibungen der Konstruktion des stofflichen Artefakts und oft auch Schaltpläne mit der Hardwarekonfiguration und den enthaltenen Elektronikkomponenten. Das sich in der Interaktion ausdrückende Verhalten des prozessierenden Artefakts wird oft in einem Video als Teil des How-tos demonstriert. Oder Fotos zeigen einzelne Zustände. Auch in einem Video kann nur eine Momentaufnahme gezeigt werden, eins von unendlichen möglichen Interaktionsmustern. Um die konkrete Seite des Digitalen Mediums und damit den Zusammenhang zwischen abstrakter Beschreibung und konkreter Realisation als auch dessen Wesen als Unvollkommenes erfahren zu können, muss das im How-to beschriebene Artefakt realisiert werden.

Umso relevanter wird es für Nachbauende und Interagierende, ein Verständnis der programmierten Abstraktionen zu entwickeln, um solche Artefakte ‚durchschauen' und kompetent nutzen zu können. So fordert Heidi Schelhowe (2007, 2012), dass Bildungssoftware auf die dahinter liegende Abstraktion verweisen muss.

Gängige How-tos von Arduino-Projekten enthalten fast immer den zugehörigen Programmcode. Diesen können Nachbauende kopieren und so direkt für ihr Artefakt übernehmen. Somit verstecken How-tos die sich hinter dem konkreten wahrgenommenen Objekt befindenden Abstraktionen nicht, sondern legen das formale Abstrakte dar. Jedoch ist Arduino-Code im Vergleich zu anderen Sprachen für Programmier-Anfänger_innen relativ komplex und auf den ersten Blick wenig verständlich (Blikstein, 2013). Daher ist anzunehmen, dass der abgebildete Code beim Nachbauen eines Projekts dazu verleitet, ihn zu kopieren, er aber nicht immer verstanden wird und so nicht unmittelbar mit dem Verhalten des Artefakts in Verbindung gebracht werden kann.

Die obige Betrachtung macht auch deutlich, dass das Wesen Digitaler Medien eingehender betrachtet werden muss, um den Fragen dieser Arbeit nachzugehen. Deshalb befasst sich das nächste Kapitel eingehender mit dem Entwickeln – Modellieren – informatischer Systeme.

2.6 Zusammenfassung und Schlussfolgerungen

Dieses Kapitel hat das Selbermachen stofflich-digitaler Artefakte im Kontext der so genannten Makerbewegung situiert. Um dieses Phänomen zu charakterisieren und den Kontext dieser Arbeit einzugrenzen, wurden die Menschen, ihre Aktivitäten, Motivationen und Kommunikationsformen und -orte betrachtet.

Maker_innen wurden definiert als Amateur_innen, die stofflich-materielle Dinge selber schaffen aus Freude am Selbermachen und sich dabei digitaler Technologien bedienen, die ein wesentlicher Bestandteil ihrer Aktivitäten sind. Neue Fertigkeiten eignen sie sich durch eigenes Konstruieren und durch den Austausch mit anderen – insbesondere in Online-Communitys – an. Das Maker-Phänomen wird getragen von technischen Entwicklungen: Zum einen der Entwicklung des Internets, welches Vernetzung und Austausch von digitalem Material erlaubt und durch das die Makerbewegung erst existieren kann. Digitale Medien sind dabei in Form von Web2.0-Applikationen als Kommunikations- und vor allem als Informations- und Lernmedium, aber auch als Ausstellungsplattform beteiligt. Dadurch wird traditionelles *Heim*werken ergänzt und bereichert durch *Netz*werken und wird zum Making. Erst durch den Gebrauch dieser Medien wird diese DIY-Bewegung sichtbar und erst durch sie existiert und lebt sie und wird zu dem, was als Makerbewegung bezeichnet wird. Zum anderen hat genau diese Entwicklung zu weiteren technischen Entwicklungen geführt, indem sich viele Menschen vernetzen und gemeinsam Open Software und Hardware entwickeln, wie z.B. Arduino, und diese für andere Maker_innen zugänglich machen. Denn Digitale Medien werden im Makerkontext nicht nur (meist als Software) als Werkzeuge benutzt, mit dem stoffliche Objekte programmiert werden oder digitale Modelle als Vorlagen für stoffliche Objekte erstellt werden. Digitale Medien in Form von interaktiven stofflich-digitalen Artefakten sind auch selbst Ziel des Making. Auch Amateur_innen können mit den zur Verfügung stehenden Ressourcen in ihrer Freizeit zu Erfinder_innen und Entwickelnden einzigartiger stofflich-digitaler Artefakte werden und benutzen Digitale Medien damit nicht nur als Werkzeuge. Auf diese Form des Making – das Selbermachen stofflich-digitaler Artefakte – konzentriert sich diese Arbeit.

Der Makerbewegung wird mitunter große gesellschaftliche und wirtschaftliche Bedeutung beigemessen. Es wird behauptet, sie könne zur Demokratisierung von Erfindung und Produktwissen führen, da sie Partizipation vieler ermöglicht durch die geringen Einstiegshürden sowie einfache Vermarktungsmöglichkeiten. Aber auch die Tatsache, dass Menschen durch Selbermachen generell zusammengebracht werden und dies ggf. zu mehr sozialem Engagement führen kann, wird betont. Maker_innen treffen sich nicht nur im Internet und in den vergangenen Jahren erfreuen sich Orte wie Hackerspaces oder FabLabs wachsender Beliebtheit. Auch die TechKreativ-Workshops können aufgrund der Rahmenbedingungen als ein konkreter Ort für vorübergehende oder angehende Maker_innen gesehen werden. Mit den TechKreativ-Workshops wur-

de die Zielgruppe dieser Arbeit auf jene jugendlichen Amateur_innen eingegrenzt, die mit Arduino-Technologie stofflich-digitale Artefakte erfinden und konstruieren.

Jenseits von Ortsgebundenen Workshops sind How-Tos und Erklärvideos eine wesentliche – wenn nicht *die* – Austauschform unter Maker_innen. Sie kennzeichnen, wie kommuniziert wird, wie Projekte dokumentiert werden und wie neue Fertigkeiten erworben werden. Auf konkrete und allgemeinverständliche Weise werden in How-Tos oder Videos anhand praktischer konkreter Beispiele Vorgehen und Sachverhalte dargelegt. Sie dienen ihren Autor_innen als Tagebuch, Portfolio und Möglichkeit zur Kontaktaufnahme und damit zum Wissensaustausch mit Gleichgesinnten und Sichtbarwerden in der Community. Das Erstellen eigener How-Tos und Videos erfordert von ihren Autor_innen, den Prozess zu reflektieren, ihn in seine wesentlichen Bestandteile zu gliedern und anschaulich mit den jeweils am besten geeigneten Darstellungsmitteln abzubilden. Die Rezipient_innen wiederum können durch How-Tos oder Erklärvideos zielgerichtet Fertigkeiten erwerben mittels eines konkreten, praxisnahen Zugangs, der sie befähigen kann, eigene Projekte erfolgreich durchzuführen.

Wie sich Selbermachende neue Fertigkeiten und Konzepte aneignen wurde mit erfahrungs- und handlungsorientierten Lerntheorien von Dewey, Vygotsky und Papert in Verbindung gebracht. So ermöglichen How-Tos und Erklärvideos informelles Lernen durch Peers, indem sie Anleitung und Hilfestellung geben zur Konstruktion eines persönlich bedeutsamen Reflexionsobjekts, anhand dessen die diesem innewohnenden Konzepte erfahren werden können. Um auch etwas über dessen Eigenschaften als programmiertes, interaktives Artefakt zu lernen, muss dieses nachgebaut werden. Denn erst durch seine Realisation kann der Zusammenhang zwischen abstrakter, formaler Grundlage und konkretem Interface umfassend erfahren werden.

Ohne anwesende Tutor_innen werden Kommunikationsmöglichkeiten, Lernquellen und Hilfestellungen, wie sie das Internet bietet, für Selbermachende wichtiger. Gleichzeitig lebt die Maker-Community davon, dass Menschen ihre Projekte austauschen. Um also auch während der Konstruktionsprozesses voneinander lernen zu können, aber auch um eigene Projekt weiterzugeben oder wieder aufzunehmen, bräuchte es eine entsprechende Dokumentationsform. Um z.B. vernetzt Hilfe zu bekommen, muss der Stand des Projekts kommuniziert werden können. Ein in der Freizeit entstehendes Projekt wird oft unterbrochen und eine Dokumentation über den aktuellen Projektstand kann nicht nur für andere, sondern auch für einen selbst hilfreich sein. Hier setzt die vorliegende Arbeit an.

How-Tos haben sich offenbar als Dokumentationsform in der Maker-Öffentlichkeit etabliert und werden für verschiedene Arten von Selbermachprojekten erstellt. Aber How-Tos dokumentieren nur das erfolgreich zu Ende geführte Projekt. Es bleibt zu überlegen, ob und wie How-Tos verbessert werden können, so dass diese für das Abbilden von Zuständen noch nicht abgeschlossener Projekt geeignet sind. Auch wäre wünschenswert, dass abstrakte Sachverhalte, insbesondere Programmcode, nicht nur

kopierbar für das eigene Projekt mitgegeben werden, sondern die abstrakten Modelle hinter dem Verhalten eines Artefakts verständlich und erfassbar für weniger erfahrene Amateur_innen dargestellt werden, damit sie ihn für eigene Zwecke weitermodifizieren oder verbessern können. Da das Erstellen von Dokumentationen eine gewisse Abstraktionsfähigkeit gegenüber des Prozesses bedarf, aber auch einen gewissen Aufwand bedeutet, bleibt ferner zu überlegen, wie auch Anfänger_innen brauchbare kompakte Dokumentationen bzw. Anleitungen unkompliziert und praxis-orientiert erstellen können.

An dieser Stelle stellt sich auch die Frage, welche Eigenschaften eines noch nicht fertigen Artefakts abgebildet werden sollten und mit welchen allgemeinverständlichen Mitteln dies geschehen kann, um das Projekt auch für Anfänger_innen nachvollziehbar dazustellen. Um dem nachzugehen, sollen in den folgenden Kapiteln die *Modelle*, die beim Making entstehen, als auch der Modellierungsprozess eingehend betrachtet werden. Dazu wird im nächsten Kapitel eine theoretische Herangehensweise zur Beschreibung selbstgemachter stofflich-digitaler Artefakte und deren Erstellung entwickelt. In diesem Zuge wird auch das Wesen Digitaler Medien als Modellierte genauer betrachtet.

Kapitel 3

Informatische Modellbildung

Bei Artefakten, die beim Making und in TechKreativ-Workshops entstehen, handelt sich um Gegenstände, die zwar Computereigenschaften besitzen, aber eine andere stoffliche Form und andersartige Interfaces als herkömmliche Computer haben (sie erscheinen z.B. wie nicht-digitale Alltagsobjekte). Anstatt durch Maus oder Tastatur erfassen sie Interaktionen und Umgebungsinformationen durch Sensoren, verarbeiten diese und wirken durch Aktuatoren – anstatt durch einen Bildschirm – auf ihre Umwelt und die Interagierenden zurück. Damit können solche Artefakte den Bereichen *Physical Computing* (O'Sullivan & Igoe, 2004) und *Tangible User Interfaces* (TUIs) zugeordnet werden (siehe auch Dittert, Katterfeldt & Reichel, 2012). Physical Computing bezeichnet ein Sensor-basiertes Erfassen und Aktuator-basiertes Kontrollieren der stofflichen Welt durch ‚Computing'. Die involvierten Computer entsprechen nicht dem, was wir als PCs kennen, sondern können verschiedene, dem Bedarf und der Interaktion angepasste, stoffliche Formen haben (O'Sullivan & Igoe, 2004, S. xvii). TUIs bezeichnen ebenfalls Interfaces, die digitale und stoffliche Welt miteinander verbinden (Shaer & Hornecker, 2009). Kerndisziplin des Physical Computing und des Erstellens von TUIs – und damit des Selbermachens physisch-digitaler, programmierter Artefakte – ist die Informatik (engl. Computer oder Computing Science). Um sowohl den Entwicklungsprozess als auch die Repräsentationen, die dabei entstehen, zu beschreiben und zu charakterisieren, wird in diesem Kapitel eine theoretische Sichtweise eingenommen, die sich an der Vorstellung des Erstellens informatischer Systeme als Modellbildung orientiert.

Die Idee von informatischen Systemen als modellierte (subjektiv wahrgenommene) Ausschnitte von Wirklichkeit wurde in den 1990er Jahren vermehrt thematisiert. Dabei gilt Modellbildung als eine zentrale Tätigkeit der Informatik. Wenn Informatiker_innen beispielsweise Software entwickeln, steht diese als ein Modell für etwas (Schinzel, 2001). Die theoretische Herangehensweise über eine Modelltheorie hat den Vorteil, dass mit ihr nicht nur der Entwicklungsprozess – die Modellbildung – beschrieben werden kann. Sie eignet sich auch, um die dabei entstehende Manifestierungen des

Vorgehens – die Modelle – einzuordnen und zu analysieren. Darüber hinaus ist dieser Ansatz nicht nur im Informatikkontext anwendbar, sondern lässt sich durch seine Allgemeinheit auch auf andere Fachgebiete, die am Erstellen stofflich-digitaler Artefakte beteiligt sind, übertragen (siehe Kapitel 4).

3.1 Die Allgemeine Modelltheorie (AMT)

Um informatische Modellbildung zu beschreiben, soll hier, wie schon in anderen Arbeiten zu diesem Thema (z.B. Goorhuis, 1994; Schütte, 1999; Kaschek, 1999; Thomas, 2002), auf die „Allgemeine Modelltheorie“ (AMT) des Philosophen und Mathematikers Herbert Stachowiak (1973) zurückgegriffen werden. Sie definiert ein Modell als ein reduziertes Abbild eines der Realität entstammenden Originals. Ein Modell wird von einem Subjekt zu einem bestimmten Zweck und in einem bestimmten Kontext geschaffen. Aufgrund dieser breiten Definition ist die AMT anwendbar auf Modelle jeglicher Art.

Die AMT geht davon aus, dass Modelle erstellt werden, wenn ein Original schlecht zugänglich (kognitiv und/oder stofflich) ist, sei es aufgrund seiner Größe, seines Ortes oder seiner Komplexität und Unübersichtlichkeit. Modelle dienen der Veranschaulichung von Zusammenhängen als Darstellungsmodelle, zum Ermitteln von Hypothesen als Experimentalmodelle, sie bündeln als theoretische Modelle Erkenntnisse oder helfen als operative Modelle bei Entscheidungen und Planung. Dabei können sie zugleich Modell von einer Idee, aber auch Modell für ein Produkt sein (Stachowiak, 1973).

3.1.1 Modellmerkmale

Nach der AMT besitzt ein Modell Attribute (Eigenschaften und Relationen) und definiert sich durch seine drei Hauptmerkmale „Abbildungsmerkmal“, „Verkürzungsmerkmal“ und „Pragmatisches Merkmal“ (Stachowiak, 1973, S. 131f.).

Abbildungsmerkmal

Das Abbildungsmerkmal beschreibt, dass Modelle natürliche oder künstliche Originale repräsentieren bzw. abbilden (im mathematischen Sinne), also „Modelle *von etwas*“ (Stachowiak, 1973, S. 131) sind. Dabei können die Originale stofflich, symbolisch oder auch nur gedanklich vorhanden sein und auf natürliche Art oder durch technische Herstellung entstanden sein. Demzufolge können auch Modelle als Originale fungieren. Mit dem Abbildungsbegriff beruft sich Stachowiak auf eine mathematisch-mengentheoretische Definition, wonach das Modell z.B. eine Teilmenge der Attribute des Originals enthält (Stachowiak, 1973).

Verkürzungsmerkmal

Das Modell ist eine verkürzte Abbildung seines Originals, d.h. nur Attribute (oder Prädikate), die den Modellierenden relevant erscheinen, finden sich im Modell wieder.

Pragmatisches Merkmal

Modelle werden für einen bestimmten Zweck von und für Subjekte in sozialen Kontexten erstellt. Dadurch sind sie nicht nur Modelle von etwas, sondern auch für jemanden zu einem bestimmten Zweck in einem bestimmten Zeitraum (Stachowiak, 1973). Als Subjekt können auch mehrere Beteiligte oder ein System gelten. Diese Betonung des Subjekts als eines von drei Hauptmerkmalen zeigt die Verortung der AMT im Systematischen Neopragmatismus. Stachowiaks Modellbegriff ist also im Pragmatismus einzuordnen. Modelle sieht er als „Konstruktion von Wirklichkeit“ (Stachowiak, 1983, Buchtitel) an.

Die „Allgemeine Modelltheorie“ (AMT) von Stachowiak (1973) fasst den Begriff des Modells sehr weit und kann somit auf Modelle verschiedener Art aus verschiedenen Disziplinen angewandt werden, so dass fast alles Geschaffene, was uns umgibt, unsere Kommunikationsmittel und gedanklichen Repräsentationen, als Modell aufgefasst werden kann, abhängig von der jeweiligen Perspektive. Jedoch beruht die AMT auf einer formalen, mathematischen Definition der Mengenlehre, um das Abbildungsverhältnis zwischen Original und Modell zu beschreiben. Auf diese beruft sich Stachowiak auch bei weiteren Charakterisierungen wie der „Präterition“ (Stachowiak, 1973, S. 156) und der „Abundanz“ (Stachowiak, 1973, S. 156) von Modell-Attributen des Originals: Finden sich im Modell weniger Attribute als im Original, so wird von präterierten Attributen gesprochen. Modelle können aber auch Attribute enthalten, die nicht im Original zu finden sind und z.B. erklärende Funktion haben. In diesem Fall handelt es sich um abundante Attribute. Des Weiteren können Originalattribute auch im Modell umgedeutet werden, so dass sie eine neue Bedeutung erhalten (vgl. Stachowiak, 1973, S. 155f.).

Neben der quantitativen Beurteilung des Verhältnisses zwischen Modell und Original betrachtet Stachowiak auch das qualitative Verhältnis zwischen Original und Modell. Er unterscheidet zwischen strukturell-formaler Angleichung und inhaltlich-materialer Angleichung (vgl. Stachowiak, 1973, S. 141ff.). Bei einer maximalen strukturellen, formalen Angleichung an das Original wird von „Isomorphie“ (Stachowiak, 1973, S. 143) gesprochen. Hierzu gehören z.B. physikotechnische Modelle (s.u.), bei denen die geometrischen Eigenschaften des Originals erhalten bleiben. Bei der inhaltlich-materialen Angleichung bleiben die Materialität und damit die Bedeutung erhalten, es erfolgt keine Umkodierung. Im Falle maximaler Angleichung wird von einem „isohylen“ (Stachowiak, 1973, S. 171) Modell gesprochen.

Welche der genannten Qualitäten ein Modell besitzt und wie diese voneinander abgegrenzt sind, hängt von Betrachter und Zweck, also pragmatischen Faktoren, ab,

so dass ein Modell mitunter aus zweierlei Sichtweisen – der Urheber_innen und der Nutzenden des Modells – beurteilt werden kann. Ein Modell, das sowohl isohyl als auch isomorph zu seinem Original ist, kann als Kopie des Originals angesehen werden.

Da Originale auch schon Modelle sein können, können Modelle von Modellen erstellt werden, so dass in einem Entwicklungsprozess eine Reihe von auseinander hervorgegangenen Modellen entstehen können. Dies wird besonders deutlich am Beispiel der Informatik, in der meist im Laufe eines Modellbildungsprozesses, z.B. beim Software Engineering, Ketten von Modellen konstruiert werden (Thomas, 2002), worauf weiter unten ausführlich eingegangen wird.

3.1.2 Modellkategorien

Die AMT unterscheidet die Modellkategorien „graphische Modelle“ (Stachowiak, 1973, S. 159) und „technische Modelle“ (ebenda) sowie „semantische Modelle“ (Stachowiak, 1973, S. 196) (siehe auch Abbildung 3.1).

Graphische Modelle

Graphische Modelle sind zweidimensionale, visuelle deskriptive Darstellungen (auch imaginäre) zu Visualisierungszwecken. Beispiele sind erinnerte Bilder oder auch Visualisierungen abstrakter Beziehungen. Stachowiak (1973, S. 159ff.) unterteilt sie in „Bildmodelle“ (Stachowiak, 1973, S. 163) mit ikonischem Charakter wie z.B. Fotografien oder Gemälde und „Darstellungsmodelle“ (Stachowiak, 1973, S. 165) mit schematischem, eher nicht-ikonischem Charakter, deren Verständnis die Kenntnis der verwendeten Zeichen voraussetzt und die sich folglich von ihrem Original entfernt haben, wie z.B. Strukturformeln oder Flussdiagramme. Übliche diagrammatische Modelle im Software Engineering sind beispielsweise als symbolische Darstellungsmodelle einzuordnen (vgl. Thomas, 2002, S. 60).

Technische Modelle

Technische Modelle definiert die AMT als „(...) raum-zeitliche und materiell-energetische Repräsentationen von Originalen, die von beliebiger Natur sein können.“ (Stachowiak, 1973, S. 174) Dreidimensionale, materielle Objekte können daher meist den technischen Modellen zugerechnet werden.

Technische Modelle benutzen Symbole, um Eigenschaften bzw. Attribute ihrer Originale abzubilden. Sie sind damit mitunter nicht so ikonisch und allgemeinverständlich wie Bildmodelle. Eine erste Unterkategorie bilden die physikotechnischen Modelle. Diese wiederum unterteilt die AMT in mechanische und elektrotechnische Modelle (vgl. Stachowiak, 1973, S. 190). Mechanische Modelle sind statisch oder dynamisch. Sie sind dynamisch, wenn Zeit als ein veränderlicher Parameter als ein Attribut im

Modell vorkommt. Ein Beispiel für ein statisches mechanisches Modell ist ein Globus. Auch repräsentieren zahlreiche statische Modelle Konstrukte rein gedanklicher Art, als „raumzeitlich-materiell noch zu Verwirklichendes“ (Stachowiak, 1973, S. 176). Elektrotechnische Modelle umfassen z.B. elektromechanische Modelle, elektronische Modelle wie Computer, Simulationsmodelle und elektrochemische Modelle. Neben der Unterkategorie der physikotechnischen Modellen unterscheidet die AMT bio-, psycho- und soziotechnische Modelle, wie z.B. Tierversuche in der Medizin oder repräsentative Gruppen von Menschen (Stachowiak, 1973).

Semantische Modelle

Bei den semantischen Modellen handelt es sich um das Metamodell eines Kommunikationssystems mit semantischen Stufen. Die erste Stufe bilden die Zeichenträger als materielle Information, die zweite Schicht beinhaltet Bewusstseinsfunktionen, d.h. interne gedankliche Gebilde. Diese Schicht verbindet wiederum die erste mit der dritten Stufe der semantischen Informationen als bedeutungstragende Zeichen oder Kombinationen von Zeichen. Außerdem ergänzt Stachowiak noch eine nullte Stufe der „Ausdrucksatome“ (Stachowiak, 1973, S. 200) mit hilfswissenschaftlicher Funktion. Neben Sprache gehören zu den semantischen Modellen auch formale Modelle, wie beispielsweise in Mathematik und formaler Logik (Stachowiak, 1973).

Dieter Müller beruft sich in seiner Dissertation „Simulation und Erfahrung: Ein Beitrag zur Konzeption und Gestaltung rechnergestützter Simulatoren für die technische Bildung“ (Müller, 1998) ebenfalls auf die AMT. In seiner Arbeit führt er eine Kategorie der „textuellen Modelle“ (Müller, 1998, S. 80) ein, die zu den semantischen Modellen gehören. Diese seien Zeichenmodelle und ähnlich wie die graphischen Modelle von der Interpretation bzw. von dem Wissen um die Bedeutung der verwendeten Zeichen abhängig (Müller, 1998).

> „Das Charakteristikum dieser Modelle besteht darin, daß Elemente, Beziehungen und Eigenschaften von Originalen durch Sprachzeichen zum Ausdruck gebracht werden. Dabei existiert keine materiale Ähnlichkeit zwischen den Attributen des Modells und denen des entsprechenden Originals. Das Zeichenhafte dieser Modelle setzt auch eine inhalts- und bedeutungsmäßige Angleichung an das Original *nicht* voraus. Als Zeichen für eine Originalgegebenheit können verschiedenste Sprachsymbole auftreten, die jeweils aus pragmatischen Gründen gewählt werden.“ (Müller, 1998, S. 89)

Er unterscheidet des Weiteren zwischen nicht-formalen und formalen textuellen Modellen. Die nicht-formalen Modelle basieren auf natürlichen Sprachen und werden weiter unterteilt in „umgangssprachliche Modelle und fachsprachliche Modelle“ (Müller, 1998, S. 81), die der dritten und vierten Stufe der semantischen Modelle zugeordnet werden können (vgl. Müller, 1998, S. 80ff.).

Anderes gilt für formale textuelle Modelle:

> „Formale sprachliche Darstellungsformen hingegen basieren auf künstlichen Symbolsprachen, die syntaktisch und semantisch vollkommen geregelt und für spezifische Zwecke konstruiert sind." (Müller, 1998, S. 81)

Als Unterkategorien dieser Modelle werden „programmiersprachliche Modelle" (Müller, 1998, S. 81) und „formal-mathematische Modelle" (Müller, 1998, S. 81) genannt, die ab der fünften semantischen Stufe der AMT einzuordnen sind.

Nicht-formale Modelle erfüllen demnach oft als Attribute eine „Abundanzfunktion" (Müller, 1998, S. 82), wenn sie in Kombination mit anderen Modelltypen (graphischen, technischen oder auch formalsprachlichen) auftreten, indem sie z.B. zum besseren Verstehen des Modells beitragen wollen. Als Beispiele sind Kommentare im Programmcode, Legende und Beschriftung von Elementen in einem Diagramm oder Ländernamen auf einem Globus denkbar (Müller, 1998).

Da die Kategorie der textuellen Modelle für den vorliegenden Fall geeigneter erscheint, um später Modelle zu analysieren und einzuordnen, als die komplexe, philosophisch argumentierende Stufenunterscheidung der semantischen Modelle der AMT, wird die Modellgruppe der textuellen Modelle von Müller (1998) für diese Arbeit übernommen neben den graphischen und den technischen Modellen der AMT (siehe Abbildung 3.1).

3.1.3 Einordnung der AMT

Mit der Allgemeinen Modelltheorie stellte Herbert Stachowiak (1973) eine Modelltheorie ausgehend vom Neopragmatismus und dem System- und Modellverständnis der Kybernetik[1] auf. Sie wird dem Systematischen Neopragmatismus zugeordnet (Stachowiak, 1983). Dieser geht davon aus, dass es keine objektive Erkenntnis gibt, sondern diese durch das Subjekt und dessen praktisches Handeln hergestellt wird, das auf einen Zweck ausgerichtet ist. Auch Denken oder Erkennen gelten dabei als Handlung. Es kann also nicht von einer objektiven Beschreibung von Wirklichkeit ausgegangen werden, sondern Wahrheit wird konstruiert (Hochkeppel, 1989). Erkenntnis ist somit an Subjekte und deren Absichten, Zwecke und Ziele gebunden und kann nur innerhalb dieser gültig sein (Stachowiak, 1989a). Auch kann (wissenschaftliche) Erkenntnis nicht losgelöst von den Theorien betrachtet werden, in die sie eingebettet ist. Wenn also im Folgenden das Attribut ‚pragmatisch' benutzt wird, so ist damit nicht die umgangssprachliche Bedeutung gemeint, sondern die philosophische Denkrichtung des Neopragmatismus.

Modelle sind nach Auffassung der AMT Konstruktionen, die von und für Subjekte zu einem bestimmten Zweck in einem bestimmten Zeitraum anhand von Originalen gebildet werden. Das Original soll hier nicht als objektive Wirklichkeit – die es

[1] Diese befasst sich mit der Steuerung selbst regelnder Systeme und deren Rückkopplungsmechanismen bzw. Informationsfluss (Stachowiak, 1989b).

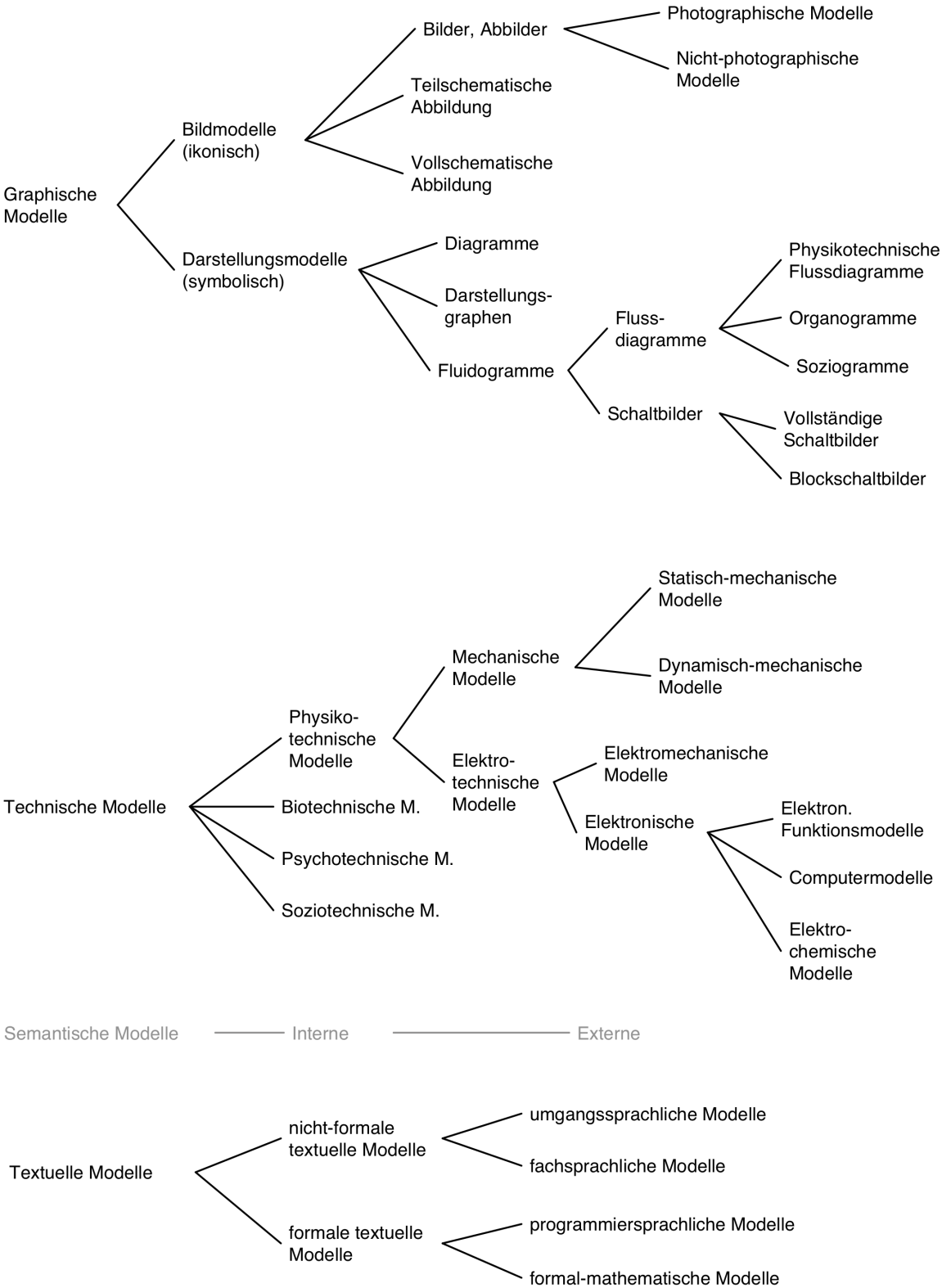


Abbildung 3.1: Modellkategorien der AMT mit Untermodellen nach Stachowiak (1973, S. 168ff.) und Müller (1998, S. 81)

im Sinne des Neopragmatismus auch nicht geben kann – oder philosophisches passives Objekt gelten, sondern als die Ausgangslage eines Modellbildungsprozesses, in dem ein Subjekt ein Modell konstruiert. Auch wenn von modellierter Realität oder Realitätsausschnitten im Folgenden gesprochen wird, so müssen diese Originale als wahrgenommene Realität (und somit auch schon als ggf. mentales Modell) angesehen werden.[2]

Die AMT liefert eine grundlegende formale Beschreibung der Beziehung zwischen Modell, Original und Subjekt, die auf nahezu alle (alltagssprachlichen) Modelle und kulturell entstandenen Artefakte anwendbar ist. Sie hat sich als geeignet für die Beschreibung von Modellen in der Informatik erwiesen, da sie auch auf die Vielfalt der Modelle, die in der Informatik vorkommen, anwendbar ist (wie gezeigt von Thomas, 2002). Demzufolge wird sie häufig zur Beschreibung von Modellen in der Informatik herangezogen (z.B. Goorhuis, 1994; Schütte, 1999; Kaschek, 1999; Thomas, 2002).

Diese Perspektive ist insofern interessant, als es sich bei der AMT um ein sehr allgemeines Konzept handelt, das nicht nur auf die Informatik bezogen werden kann. Somit lässt es sich auch auf andere Bereiche im HCI-Kontext anwenden, was in Kapitel 4 geschehen wird. Das Konzept bietet sowohl eine abstrakt-theoretische wie auch eine konkret-praxisorientierte Erschließung und Betrachtung der Thematik an. Die AMT lässt sich damit auf Making übertragen. Mit diesem Zugang kann deshalb der Frage, wie das Abbilden von Modellen stofflich-digitaler Artefakte für Amateur_innen unterstützt werden kann und wie diese beschrieben werden können, nachgegangen werden.

So soll die AMT auch in dieser Arbeit theoretische Grundlage für die Betrachtung des Modellierens von Informatiksystemen dienen, die als stofflich-digitale Artefakte ‚selbstgemacht' werden.

3.1.4 Begrifflichkeiten

Der Begriff des *Modells* wird in dieser Arbeit durch Bezugnahme auf die AMT sehr weit gefasst, wie oben dargelegt wurde. Modell beschränkt sich hier weder auf mathematisch-logische Modelle, noch sind damit im Informatikkontext nur Simulationsmodelle gemeint, für die das Wort Modell in der Literatur oft benutzt wird. Auch sind damit hier nicht nur Modellierungssprachen wie z.B. die Unified Modeling Language (UML)[3] oder innerhalb des objekt-orientierten Paradigmas vorkommende Modelle gemeint. Gleiches gilt für die Begriffe *Modellierung* und *Modellbildung*. Diese Begriffe gelten somit auch für Modellierungsprozesse in anderen Kontexten, z.B. im Design (siehe Abschnitt 4.4). Mit *informatischer* Modellbildung oder Modellierung ist hier explizit das Modellieren eines Informatiksystems gemeint, bei dem es sich meist um konzeptuelles Modellieren (s. Abschnitt 3.2.3) handelt.

[2] Siehe dazu auch die Kontroverse zwischen Kaschek (1999); Schütte (1999); Kaschek (2000); Schütte (2000) zum Verhältnis von Realität, Original und Abbildung.

[3] `http://www.uml.org` (aufgerufen am 02.01.2013)

Das Erstellen von informatischen Systemen bzw. Digitalen Medien wird in dieser Arbeit als *informatische Modellbildung* bezeichnet. *Informatiksystem* oder *informatisches System* wird für Systeme verwendet, die programmierbar sind. Charakteristisch für diese Informatiksysteme ist, dass sie Daten binär bzw. digital verarbeiten. Bei einem solchen System kann es sich sowohl um Software handeln, die auf beliebigen Hardwarekonfigurationen läuft, als auch Computerprogramme, die nur für eine eigens konstruierte, nicht handelsübliche Hardwarekonstellation entwickelt wurden und nur im Zusammenspiel mit dieser benutzbar sind. Damit wird die Kombination aus Hardware und Software als *ein* System bzw. Modell bezeichnet, da beides nur in Kombination als Modell wirken kann bzw. als solches konzipiert wird. Stofflich-digitale ubiquitäre Informatiksysteme, die aus kleinen Hardware-Software-Einheiten bestehen, werden explizit miteingeschlossen.

Der Begriff des *Konstruierens* oder der Konstruktion wird in dieser Arbeit häufig synonym zu basteln, bauen, (selber) machen oder auch modellieren gebraucht. In der Literatur wird Konstruieren allerdings auch benutzt, um die planerische Tätigkeit von Ingenieur_innen oder Informatiker_innen zu beschreiben und sie einem Basteln (Pepper, 1992) oder spielerisch-gestaltenden Tätigkeiten (Nake, 2003) entgegenzusetzen. Dennoch benutzt diese Arbeit bewusst den Begriff des Konstruierens, um das Selbermachen stofflich-digitaler Artefakte zu beschreiben und bezieht sich damit auf eine konstruktivistische Prägung: Zum einen betont der Begriff die Eigenschaft von Modellen als von Subjekten Konstruierte (siehe Abschnitte 3.1.1 und 3.3). Zum anderen verweist er auf den Bezug des ‚Making' zur Lerntheorie des Konstruktionismus von Papert (1980) (siehe Abschnitt 2.5.3).

Auch der Begriff *formal* wird in dieser Arbeit mitunter anders gebraucht, als zuweilen in Informatik und Mathematik zu erwarten wäre. ‚Formal' wird im Informatikkontext meist mit formalen Sprachen assoziiert. So werden auch formale Modelle mit formalen Sprachen erstellt:

> „Eine *formale* Sprache (oder kurz *Sprache*) L über dem Alphabet V ist eine Teilmenge von V^* : $L \subseteq V^*$. Ihre Elemente heißen *Sätze* (in der mathematisch orientierten Literatur meist *Wörter*)." [Hervorhebungen im Original] (Rechenberg, 1999, S. 90)

Wobei es sich beim Alphabet um eine endliche Zeichenmenge handelt.

In einer formalen Sprache ausgedrückte Formulierungen folgen Regeln, die in einer (ebenfalls formal definierten) Grammatik festgelegt sind. Auch ihre Semantik ist mathematisch-logisch eindeutig definiert. Entsprechend können Modelle aus formalen Sprachen verifiziert werden (Floyd & Züllighoven, 1999).

Da sich diese Arbeit im Makerkontext bewegt und sich gezielt mit Amateur_innen mit wenig Vorerfahrung in Informatik und/oder Mathematik beschäftigt, wird der Begriff ‚formal' hier relativ und dem Zweck angepasst benutzt. Als formal werden demzufolge auch Modelle bezeichnet, die beim Programmieren oder beim Erstellen von Schaltkreisen entstehen und sich im Kern an formalen Regeln orientieren. Das bedeu-

tet, dass vernachlässigt wird, ob diese Modelle der Verifizierbarkeit und Eindeutigkeit standhalten. Der Begriff *unformal* wird gebraucht, um Darstellungen und Vorgehensweisen zu bezeichnen, die sich in Modellen ausdrücken, die keiner formalen Sprache (z.B. Programmiersprache, Schaltbilder, Schaltungen oder mathematischer Ausdruck) folgen. Die Bezeichnung *informell* wurde bewusst in Zusammenhang mit Modellen vermieden, weil sie mit ihrer umgangssprachlichen Bedeutung oder mit Begriffen wie ‚informelles Lernen' verwechselt werden könnte.

3.2 Modelle in der Informatik – Informatik als Modellbildung

Die AMT wird, wenngleich sie auf alle möglichen Arten von Modellen anwendbar ist, in der deutschsprachigen Literatur oft zur Beschreibung des Wesens von Informatiksystemen herangezogen, insbesondere, wenn deren Eingebettetsein in soziale Kontexte thematisiert wird (siehe z.B. Hesse & Mayr, 2008; Thomas, 2002; Müller, 1998; Goorhuis, 1994). Die Mathematikerin, Informatikerin und Genderforscherin Britta Schinzel (2001) weist darauf hin, dass für Modelle in der Informatik auch eine andere erkenntnistheoretische Perspektive, die von objektiver Realität ausgeht, eingenommen werden kann. Aus dieser Sichtweise ist die Beziehung zwischen Modell und abgebildeter Realität von Vollständigkeit, Korrektheit und Verifizierbarkeit bestimmt (vgl. a. Grube, 1995). Diese Arbeit, die hauptsächlich im Bereich der Human-Computer Interaction (HCI) angesiedelt ist, konzentriert sich demgegenüber auf die oben beschriebene konstruktivistisch geprägte Sichtweise, in der Modelle nicht nur Wirklichkeit abbilden, sondern schaffen, und diese Wirklichkeit wiederum zu Originalen für neue Modelle werden kann (Schinzel, 2001).

3.2.1 Modellvielfalt in der Informatik

In der Informatik spielt Modellbildung eine wesentliche Rolle und gilt als zentrale Tätigkeit. Modelle erscheinen nicht nur als Software, sondern auf verschiedenen Ebenen in verschiedenen Formen.

Der Informatikdidaktiker Marco Thomas (2002) ordnet informatische Modelle in fünf Kategorien ein: „Architekturmodelle" (Thomas, 2002, S. 51) (z.B. als theoretisches Maschinenmodell wie die Turingmaschine, oder als ein ein bestimmtes Paradigma verkörperndes Programmiermodell), „Vorgehensmodelle" (Thomas, 2002, S. 52) (z.B. objektorientierte Modellierung, Phasenmodelle), „Entwurfsmodelle" (z.B. Datenbankmodelle, Aufgabenmodelle i.S. einer Problemanalyse), „Untersuchungsmodelle" (Thomas, 2002, S. 55) (z.B. mathematische und Simulationsmodelle), sowie „Mentale Modelle" (Thomas, 2002, S. 56) (z.B. konzeptuelle Modelle). Er zeigt in seiner in der Informatikdidaktik angesiedelten Dissertation weiter, dass in der Informatik Mo-

delle aller Kategorien der AMT vorkommen.[4] Beispiele für graphische Modelle sind u.a. CAD-Darstellungen oder Diagramme. Allein die graphischen Modelle, die mit der gebräuchlichen UML erstellt werden können, deckten dabei schon zahlreiche Darstellungsmodelle der AMT ab. Technische Modelle sind vertreten z.B. als Computermodell im Sinne eines laufenden Programms oder auch als Mensch-Maschine-Kommunikation im Sinne eines soziotechnischen Modells. Semantische Modelle reichen von materiellen Informationen als Zeichenträger wie Signale und Daten stufenweise bis hin zu externen semantischen, der Kommunikation dienenden Modellen, wie z.B. Programmiersprachen als maschinensprachliche Modelle (vgl. Thomas, 2002, S. 59ff.). Aus einer Analyse des Gebrauchs des Begriffs Modell in Vorlesungsskripten in Informatik-Fachbereichen deutscher Universitäten leitet Thomas ab, dass damit meist semantische und graphische Modelle im Sinne der AMT gemeint sind, wohingegen technische Modelle eher als Systeme bezeichnet werden. Modelle, die beim Software Engineering erstellt werden, sind nach AMT zumeist Darstellungsmodelle und softwaretechnische Modelle. Dennoch würden die im Software Engineering verwendeten Modelle nur einen Ausschnitt informatischer Modellbildung darstellen (Thomas, 2002).

3.2.2 Modelle als Ketten

Das Erstellen von Informatiksystemen bezeichnen Wedekind, Görz, Kötter und Inhetveen (1998) als „Modellieren des Modellierens“ (Wedekind et al., 1998, S. 270). Wenn beispielsweise ein naturwissenschaftliches Modell in ein maschinell effizient ausführbares Modell übertragen werden soll, werden auf mehreren Ebenen Modelle erstellt. Zunächst wird außerhalb der Informatik das naturwissenschaftliche Modell modelliert, welches anschließend von Informatiker_innen interpretiert und als Software modelliert wird. Wedekind et al. (1998) berücksichtigen dabei Informatiksysteme noch nicht als Digitale Medien wie wir sie heute kennen. Sie beschreiben diese in Wissenschafts- und Arbeitskontexten und sehen die Rolle der Informatik in der Modellbildung darin, Modelle aus Fachwissenschaften in vom Computer effizient ausführbare Modelle (z.B. Simulationen) zu formalisieren. Dies erfordert, deren wesentliche Merkmale zunächst in Zwischenschritten, d.h. Modellen, herauszuarbeiten (Wedekind et al., 1998).

Thomas nennt informatische Modellbildung „Modellieren von Modellen“ (Thomas, 2002, S. 22) um damit auszudrücken, dass in der Informatik meist Modelle konstruiert werden, die auf andere Modelle folgen und so „Modellketten“ (Thomas, 2002, S. 49) bilden. Wie er zeigt, kommen Modelle jeden Typs im Sinne der AMT in der Informatik vor (s.o.). Um ein System zu erstellen, werden mehrere, i.d.R. verschiedenartige und oft aufeinander aufbauende Modelle entwickelt, was in ein „Modellieren von Modellen“ (Thomas, 2002, S. 67) mündet.

Durch Modellbildung in der Informatik wird also nicht nur (wahrgenommene) Realität durch Modelle deskriptiv abgebildet. Dieselben Modelle dienen im Entwicklungs-

[4] Für einen umfassenden Überblick siehe Thomas (2002, S. 60ff.).

prozess auch als Vorbilder für digitale Artefakte oder für weitere Zwischenmodelle, die durch den Modellierungsprozess konstruiert werden und wiederum eine neue Realität (und damit ein neues Original) bilden.

Das gedankliche Modell von informatischen Systemen als Modelle und deren Erstellung als Modellbildungsprozesse wird auch thematisiert, um den wesentlichen Kern von Informatiktätigkeit zu erschließen und diesen zu vermitteln. So beschäftigt sich die Informatikdidaktik nach wie vor mit der Sichtweise auf Informatik als Modellbildung und betont Modellierung als wesentliche Kompetenz und damit zu verfolgende Tätigkeit im Informatikunterricht (z.B. Schubert & Schwill, 2004; Hampel, Magenheim & Schulte, 1999; Magenheim, 2001; Hubwieser, 2004; Nelles, Rhode & Stechert, 2010).[5]

Thomas (2002) betrachtet „Modellieren von Modellen“ (Thomas, 2002, S. 67) als „Kulturtechnik“ (ebenda), mit der Kultur erschlossen und produziert würde und leitet für das Modellieren von Modellen in der Informatik allgemeinbildende Relevanz ab.

Das Entwickeln von Informatiksystemen als Modellbildung im Sinne der AMT zu betrachten ist damit nicht nur geeignet, stofflich-digitale Artefakte und deren Erstellung zu beschreiben, sondern auch, um ein Bild der Informatik zu vermitteln.

3.2.3 Konzeptuelle Modellierung

Informatische Systeme beruhen auf logischen, formalen mathematischen Anweisungen. Sie können damit prinzipiell mittels logischer und formaler Herleitung entwickelt werden, so wie bei einer mathematischen Aufgabe. Zur Spezifikation werden i.d.R. formale Modellierungssprachen verwendet. Solche Modellierungsaktivitäten, die ein ausführbares, anwendbares Informatiksystem zum Ziel haben, werden als ‚konzeptuelle Modellierung‘, oder englisch „conceptual modelling“ (Thalheim, 2011) bezeichnet.

> „Ähnlich wie ein Bildhauer erst ein Modell von dem darzustellenden Objekt anfertigt (wozu ihm im Falle einer abgebildeten Person womöglich jemand ‚Modell steht‘ (sic!) und dieses dann in das endgültige Kunstwerk umsetzt, baut sich der Software-Ingenieur erst ein konzeptuelles (...) Modell des Weltausschnitts, in den das zu konzipierende Softwaresystem eingesetzt werden soll und sodann daraus das eigentliche System.“ (Hesse & Mayr, 2008, S. 379)

Hesse und Mayr (2008, S. 379) verweisen auf den Unterschied zum *konzeptionellen* Modell, das ein Konzept betreffe, anstatt eines aufzuweisen. Der Informatiker Bernhard Thalheim beschreibt konzeptuelles Modellieren auch als „engineering activity“ (Thalheim, 2011, S. 543) und setzt es gleich mit ‚engineering‘.

Wenn in diesem Kapitel weiterhin von *informatischer* Modellierung oder Modellbildung gesprochen wird, so liegt dem die Annahme zugrunde, dass konzeptuelle formalisierende Modellierung, wie sie im Software Engineering praktiziert wird, zumindest

[5] Hierbei wurde nur der deutschsprachige Raum betrachtet.

ein wesentlicher Teil davon ist. Dadurch soll das Wesen informatischer Modellierung charakterisiert und gegenüber anderen Disziplinen abgegrenzt werden.[6]

Modelle in der Softwareentwicklung als Vorbilder und Nachbilder

Modelle in der Softwareentwicklung sind zugleich Vor- und Nachbilder, d.h. sie haben präskriptiven und deskriptiven Charakter (Hesse, 2006). UML-Diagramme beispielsweise bilden einen (wahrgenommenen) Realitätsausschnitt ab und dienen zugleich als Vorbild (bzw. Spezifikation) für ein zu modellierendes, anwendbares Informatiksystem (siehe Beispiele in Hesse, 2006, S. 102). Zusammengenommen bilden sie eine Kette von Modellen.

In Verbindung mit der objekt-orientierten Systementwicklung und damit verbundenen standardisierten Modellierungssprachen wie UML hat sich z.B. das Model Driven Development in der Softwaretechnik etabliert (Hesse & Mayr, 2008, S. 379). Am Beispiel von UML-Modellen wird dort deutlich, dass Modellieren von Modellen nicht nur als chronologisch linear-sequentiell angesehen werden darf, sondern dass dies auch parallel geschehen kann, indem mehrere Modelle zur Systembeschreibung angefertigt werden, die jeweils nur bestimmte Eigenschaften (z.B. Systemarchitektur oder Interaktionsmuster) abbilden.

Programmierung und Modellierung

Wie oben erläutert wird der Begriff der (konzeptuellen) Modellierung im Kontext der Informatik bzw. des Software Engineering in der Regel enger verstanden, als er in dieser Arbeit und in der AMT benutzt wird. Anhand von verschiedenen, konzeptuellen Modellen wird im Laufe der Entwicklung einer Software ein Programm ‚gecodet'. Dieses wird wiederum vom Rechner in Assemblercode umgewandelt und ist Vorbild für diesen usw. Der Programmcode ist zwar auch ein Vorbild für ein weiteres, maschinell erstelltes Modell – das Programm ist deskriptiv im Verhältnis zu Anforderungsdefinition oder vergleichbaren Vormodellen – jedoch ist es das letzte Modell in der Kette, das von den Modellierenden unmittelbar erstellt wird. Damit ist es ein wesentlicher Teil des Systems, das zur neuen Realität wird.

Darüber hinaus ist die Programmierung essentieller Teil des stofflich-digitalen Artefakts, denn sie macht dieses erst zum Digitalen Medium. Das Programm ist Teil des Artefakts und liegt als implementiertes Modell vor, das von der Maschine interpretiert und ausgeführt wird und das Artefakt quasi zum Leben erweckt. Programmcode ist digital übermittelbar und kann damit direkt bei anderen Entwickelnden auf deren Geräten wirksam werden, ohne von einem Subjekt neu interpretiert werden zu müssen.

Eine Trennung in (konzeptuelle) Modellierung und Programmierung kann nicht immer eindeutig getroffen werden. Auch hier haben wir es mit Modellketten zu tun,

[6] Auch wenn nicht ausgeschlossen wird, dass in der Praxis auch für Software Engineering untypische Modelle benutzt werden.

deren hintere Glieder auch als Programm bezeichnet werden können, aber dennoch auch Modelle sind. Das vorerst ‚letzte' Modell, das den Algorithmus für das Verhalten des Systems abbildet und unmittelbar vom System interpretiert werden kann, hat zwar eine Sonderrolle, weil es das informatische System mitbegründet. Trotzdem soll in dieser Arbeit mit dem Begriff der Modellierung auch der Vorgang des Erstellens dieses Modells – sowohl als visueller als auch als textueller Programmcode und damit als Modell – eingeschlossen werden.

3.2.4 Der Nutzen von Modellen

Im Software Engineering werden verschiedene Modelle erstellt, die auch als Modellketten angesehen werden können. Oft geschieht dies mit Hilfe von Modellierungswerkzeugen, die das bewusste schrittweise Erstellen aufeinander aufbauender Modelle oder das Zerlegen in Modellsichten eines Systems unterstützen. Welchen Nutzen diese haben, soll im Folgenden betrachtet werden. Dazu werden weitere, eher praktische Gründe aufgezeigt, die für die konzeptuelle aufeinanderfolgende Bildung von Modellen von oder für Informatiksysteme genannt werden.

Modelle als Prototypen

In informatischen Modellbildungsprozessen werden oft Prototypen erstellt, die nach der AMT als Modelle bezeichnet werden können und somit Modelle in einer Folge von vielen Modellen bilden. Als Prototypen, die sie von anderen Modellen wie einer Anforderungsdefinition als Text oder einer gedanklichen Vorstellung als mentales Modell unterscheidet, sollen hier vorwiegend stofflich-anfassbare oder digitale Modelle mit einer deutlichen äußeren oder strukturellen Ähnlichkeit zum angestrebten System gelten.

Budde und Züllighoven (1990) beschreiben Prototypen im Software Engineering als operationelle, ausführbare Modelle, die als Diskussionsgrundlage und Entscheidungshilfe für die am Projekt beteiligten Stakeholder und als Erweiterung der Anforderungsanalyse z.B. durch Berücksichtigung von Nutzeraspekten, dienen. Sie sind Ausgangspunkt für weitere Prototypen. Allerdings haben Prototypen im Software Engineering z.B. nicht die Funktion materieller Prototypen in der Produktion, die als Vorbilder für eine Serienproduktion dienen können (Budde & Züllighoven, 1990).

Ein Softwareprototyp simuliert nicht nur, sondern *ist* durch die materielle Eigenschaft von Software schon ein Teil des zu erstellenden Systems, d.h. er enthält Funktionen des zu erstellenden Systems. Softwareprototypen werden benutzt, um Spezifikationen und mögliche Probleme abzuklären und um experimentelle, praktische Erfahrungen zu sammeln. Budde und Züllighoven (1990) unterscheiden folglich Prototypen, die die Anforderungsfindung unterstützen und als meist äußerliche, anschauliche Beispiele einzelner Aspekte (z.B. des User Interface) des Systems dienen. Ein solcher Prototyp

kann als ausführbare Spezifikation dienen, um offene Punkte mit zukünftigen Nutzenden abzuklären und Alternativen explorativ zu erfahren. Eine andere Form von Prototypen kann die technische Sicht auf das System modellieren und bildet dessen technische Struktur und/oder dessen Code-Struktur ab und kann so den Entwickelnden zur Exploration und zum Experimentieren hinsichtlich der technischen Umsetzbarkeit dienen. Als eine dritte Art von Prototypen im Software Engineering nennen Budde und Züllighoven (1990) Pilotsysteme, die als Vorformen des zu erstellenden Systems dienen und inkrementell zum fertigen System ausgebaut werden.

Prototypen werden oft nach *Low Fidelity* (LoFi) Prototypen und *High Fidelity* (HiFi) Prototypen unterschieden (für einen Überblick im HCI-Kontext siehe Rudd, Stern & Isensee, 1996). Aus Sicht der AMT kann diese Unterteilung Hinweise auf die Menge der Attribute geben, die vom Original im Modell abgebildet sind. Ob eher LoFi oder HiFi Prototypen geeignet sind, hängt von deren aktuellem Zweck – also von pragmatischen Erwägungen – ab.

Das Beispiel der Prototypen als Modelle verdeutlicht, dass Modelle auch der Kommunikation und Kooperation dienen, oder dazu, verschiedene Sichtweisen einzunehmen und dadurch Komplexität zu reduzieren.

Mit Modellen Komplexität reduzieren

Informatische Systeme sind komplex. Ein solches System kann in der Regel nicht vollständig in einem Modell abgebildet werden, bedingt u.a. durch dessen zeitliche und räumliche Dimensionen und damit verbundene sich ändernde Zustände (Dubberly, 2009). Auch wenn z.B. ein Programmcode das programmierte Systemverhalten abbildet, so enthält dieser keine Informationen über die Architektur der Hardware oder einer benötigten Datenbank. Unterschiedliche, sich ergänzende Modelle eines Systems helfen den Entwickelnden, dessen Komplexität zu überblicken und zu bewältigen. So gelten verschiedene, sich ergänzende Perspektiven auf ein komplexes System als förderlich für dessen Verständnis (siehe dazu Ausführungen in Faust, 2008).

Im Modellbildungsprozess für informatische Systeme werden Modelle aus verschiedenen Sichtweisen auf ein System erstellt. Damit entstehen eine Vielzahl von Modellen wie z.B. Architekturmodelle, Datenbankmodelle, Interaktionsmodelle (vgl. z.B. Thomas, 2002). Weitere typische Modelle im Software Engineering sind Prozessmodelle (z.B. als Petri-Netze), Modellierung von Abläufen mit Hilfe von Flussdiagrammen oder Flow Charts oder Datenmodellierung (z.B. mit Entity-Relationship-Diagrammen). Die verschiedenen Sichtweisen, die dabei eingenommen werden, bilden sich auch in den zur Verfügung stehenden diagrammatischen Modelltypen in der UML ab (z.B. als Interaktionsdiagramme, Strukturdiagramme, Zustandsdiagramme oder Klassendiagramme) (vgl. z.B. Seemann & Gudenberg, 2000).

3.3 Konflikte: Informatische Modelle im Kontext

3.3.1 Informatische Modellbildung als Konstruktion

Ausgehend von der AMT betrachtet auch der Informatiker und Systemtheoretiker Henk Goorhuis (1994) Entwurfsprozesse in der Informatik als Modellbildungsvorgang. Er bezeichnet diesen als „konstruktivistisch“ (Goorhuis, 1994, S. 77), um den Einfluss von Subjekten im Modellierungsprozess zu betonen, der dem Gedanken an ein objektives Abbilden von Sachverhalten durch Informatiksysteme entgegen steht. Er unterscheidet zunächst zwei Modellierungsphasen: In der kogitativen Phase erstellt das Modellsubjekt ein auf seinen Wahrnehmungen und Antizipationen beruhendes mentales Modell – seine Vorstellung – des Modellobjekts, also des zu modellierenden Sachverhalts bzw. Ausschnitts von Wirklichkeit. Dieses Modell überführt er dann in der konstruierenden Phase in ein formales Modell, so dass es in einer von Maschinen ausführbaren Form – dem Informatiksystem als finalem Modell – formuliert werden kann. Dabei verfolgt das Modellsubjekt je nach Phase verschiedene Intentionen (siehe Abbildung 3.2). Während das Modell der ersten Phase durch Wahrnehmung konstruiert wird, wird das der zweiten Phase anhand von logischen, formalen Gesetzmäßigkeiten konstruiert, die durch das Informatiksystem bedingt sind. Dies führt zu diversen Zielkonflikten, die sich auf die Beziehung zwischen Nutzenden im sozialen Kontext und Medium auswirken (Goorhuis, 1994).

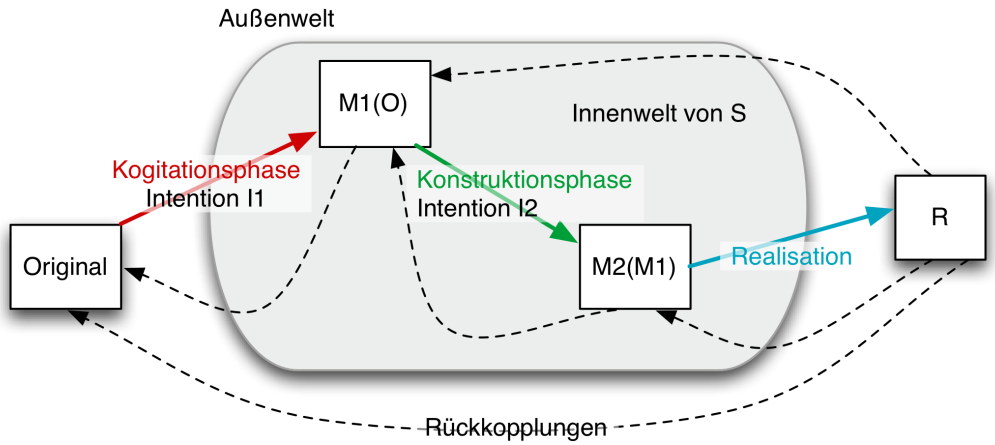


Abbildung 3.2: Phasen der Modellierung in Anlehnung an Goorhuis (1994, S. 33): M1: Modell der Kogitationssphase, M2: Modell der Konstruktionsphase, R: realisiertes Modell (Informatiksystem), S: Modellsubjekt

Das Modellieren in der Informatik ist laut Goorhuis (1994) also aufgrund der verschiedenen Bedingungen in den beiden aufeinander aufbauenden Phasen von inhaltlichen Zielkonflikten geprägt. Der Endnutzer als Subjekt wird immer eine andere Sicht als das informatisch modellierende Subjekt auf das System haben. Diese Sicht kann das modellierende Subjekt nur unzureichend antizipieren.

3.3.2 Formalisierung und Subjekt-Objekt-Spaltung

Aus Sicht der AMT überführt bei der Modellierung von Informatiksystemen ein Subjekt schrittweise nicht-formale, subjektiv wahrgenommene Wirklichkeit in ein formales Modell. Nach der AMT haben Modelle ein pragmatisches Merkmal, d.h. sie werden von einem Subjekt zu einem bestimmten Zweck erstellt. Goorhuis weist auf die Problematik der Modellschaffenden hin, die eine vermeintlich objektive Realität zu modellieren versuchen. Für informatische Modellbildungsvorgänge nennt er neun Zielkonflikte. Diese sind von der Unvereinbarkeit zwischen nicht-formaler Realität und Formalität erforderndem Zielmodell (Informatiksystem) geprägt. Exemplarisch seien hier genannt: Zielkonflikt I als die Diskrepanz zwischen Gesetzmäßigkeiten von Wahrnehmung und Logik, Zielkonflikt III als das „Abbilden einer nicht-formalen Welt in eine formale Welt“ (Goorhuis, 1994, S. 58) und Zielkonflikt VIII als das Problem der Übersetzbarkeit von Alltagssprache (durch Intuition, Unbewusstes etc. geprägt) in die formale Programmiersprache (vgl. Goorhuis, 1994, S. 54ff.).

Da der Modellierungsvorgang von diesen Zielkonflikten bestimmt ist und diese irreduzibel (d.h. unauflösbar) sind, müssen Modellierende Entscheidungen treffen. Damit konstruieren sie ihre Auffassung der zu modellierenden Wirklichkeit im Laufe der Modellierung weiter. Somit könne der Modellbildung in der Informatik ein „*konstruktivistischer* Aspekt“ (Goorhuis, 1994, S. I) zugesprochen werden und entsprechend schlägt er den Konstruktivismus als erkenntnistheoretisches Modell zu Charakterisierung vor (Goorhuis, 1994, S. II). Schon die Problemstellung wird vom Modellsubjekt konstruiert, und im Laufe des Modellbildungsprozesses sei durch den Selbstbezug nicht mehr entscheidbar, welcher Anteil der Wahrnehmung konstruiert ist (vgl. Goorhuis, 1994, S. II).

Um diese unauflösbare Problematik des Selbstbezugs bzw. der Subjekt-Objekt-Spaltung zu begegnen, schlägt Goorhuis die Entwicklung dynamischer Methoden für die Softwareentwicklung vor, weil diese keine Subjekt-Objekt-Spaltung aufwiesen. Als wesentlich sieht er es an, dass die am Prozess beteiligten sich der Subjektivität ihrer (konstruierten) Wahrnehmung bewusst werden und Bereitschaft zeigen, diese zu ändern, anstatt an starren Pflichtenheften o.ä. festzuhalten (Goorhuis, 1994).

Als eine Antwort darauf können die mittlerweile entwickelten so genannten agilen Methoden angesehen werden (siehe Abschnitt 3.3.5).

3.3.3 Wirksamwerden von Modellen in sozialem Kontext

Floyd und Klischewski bezeichnen Informatik-Modelle als „Fenster zur Wirklichkeit“ (Floyd & Klischewski, 1998, S. 21) im Sinne einer Wahrnehmungsperspektive. Bezogen auf die Entwicklung und beabsichtigte Wirkungsweise dieser Modelle schlagen sie aber den Begriff „Handgriff zur Wirklichkeit“ (ebenda) vor. Ebenfalls aus einer Perspektive von Modellbildung als Konstruktion unterscheiden sie drei Modellierungs-

ebenen: Das „Anwendungsmodell“ (Floyd & Klischewski, 1998, S. 22), das den zu modellierenden Gegenstandsbereich noch konkret (ggf. idealisiert) darstellt, das „formale Modell“ (ebenda) als die (logische) Spezifikation und das „Berechnungsmodell“ (ebenda) als ein Programm aus berechenbaren Funktionen. Bevor das eigentliche Informatiksystem mit Hilfe einer Programmiersprache implementiert werden könne, wird ein (zunächst mentales) Modell eines Realitätsausschnittes (dem Gegenstandsbereich) in ein formales Modell überführt. Diese Aktivitäten benennen sie als „Informatisieren“, „Diskretisieren“ und „Systemisieren“ (Floyd & Klischewski, 1998, S. 21f.).

Zu diesem Vorgehen geben Floyd und Klischewski deren Einbettung in soziale Kontexte zu bedenken:

> „Informatik-Modelle stehen nicht als formale Gebilde für sich genommen, sondern werden in den von uns getragenen sozialen Prozessen konstituiert“ (Floyd & Klischewski, 1998, S. 26).

(Anwendungs-)Modelle könnten daher nicht als isoliert wahrgenommen werden, sondern entspringen einem sozialen Kontext und basieren dabei auf Annahmen über Wirklichkeit, sind soziale Konstrukte. Sie würden erst durch soziale Prozesse bedeutsam und wirksam, beeinflusst von subjektiven Perspektiven. Dabei bestehe eine Diskrepanz zwischen Entstehung und Wirksamkeit (bzw. dem Wirksamwerden) der Informatikmodelle. Ein Ausgangsmodell entstehe in einem sozialen Kontext, durchlaufe eine Formalisierung und agiere schließlich, als Informatiksystem, wiederum in einem sozialen Kontext. Die Wirksamkeit entstehe dabei auf zwei Ebenen: auf mentaler Ebene durch Auseinandersetzung der Akteure mit dem Wirklichkeit abbildenden Modell und auf autooperationaler Ebene durch die direkte Auswirkung des Computerprogramms auf die Realität und damit auf den sozialen Kontext (vgl. Floyd & Klischewski, 1998, S. 24).

In dieser Erscheinungsform als Anwendungsmodell lassen sich seine Funktionsweise und somit die Intentionen der Modellierenden weniger leicht nachvollziehen als es bei nicht-digitalen Modellen der Fall ist. Zum Beispiel wird ein Text – nach der AMT ein Modell – in derselben Form vom Leser aufgenommen, in der der Autor ihn verfasst hat (vgl. Goorhuis, 1994, S. 6).

3.3.4 Selbstbezug

In der Regel ist das Ziel der übergeordneten Modellierungstätigkeit und damit des resultierenden Informatiksystems, eine geistige Tätigkeit des Menschen zu unterstützen oder gar zu ersetzen (Thomas, 2002). Doch Informatiksysteme (d.h. Modelle) können aufgrund ihrer prozessierenden Eigenschaft den sozialen Kontext, dem sie entstammen, verändern. Das formale Modell (Informatiksystem) wird wirksam in der Realität, aus der es ursprünglich entstammt. So entsteht ein Selbstbezug oder Rückbezug. Das ausführende Modell verändert seine eigene Ausgangslage, was zu Wechselwirkungen führen kann (Humbert, 2002; Thomas, 2002).

Schon 1977 forderte Carl Adam Petri[7]:

> „ (...) the existence of computer technology makes a new approach to modelling necessary and, at the same time, feasible“ (Petri, 1977, S. 435)

Er forderte, Informatik als Kommunikationsdisziplin[8] zu betrachten und erläuterte, dass das Vorgehen bei mathematischer Modellierung bzw. beim Modellieren nicht-digitaler Werkzeuge nicht auf das Modellieren von Software übertragen werden könne:

> „In the computer age, symbolic production (e.g. of software) has greatly increased. And the procedural methods generated by it are no longer mainly fed back to the human activity of thinking about useful experiments, but are rather fed directly into the activity of using computers (...): As a consequence, the final products of symbolic activities are allowed to become much more complex and ‘powerful’ than when they had to be used directly by humans only.“ (Petri, 1977, S. 436ff.)

Schon hier weist Petri also darauf hin, dass, wenngleich sich informatische Modelle formal beschreiben lassen, diese in einem ganz und gar unformalen sozialen Kontext mit ihrer Komplexität und Eigenständigkeit wirken. Dies können sie folgenreicher als nicht autonome Tools, so dass ein reflektiertes Vorgehen über die erfolgte Modellierung der Ausgangsmodelle (das zu Modellierende) im Modellierungsprozess erforderlich wird.

Der Informatikdidaktiker Ludger Humbert sieht das Problem im praktischen Vorgehen bei der Modellierung:

> „Um die mit der Methode der informatischen Modellierung verbundenen Probleme zu verdeutlichen, ist darauf hinzuweisen, dass ausgehend vom Problembereich eine Dekontextualisierung vorgenommen wird, die im Zuge des Einsatzes als Teil eines konkreten Informatiksystems eine Rekontextualisierung erfährt.“ (Humbert, 2002, S. 112)

Deshalb schlägt er die Einbeziehung von Anwender_innen in den Entwicklungsprozess im Sinne einer partizipativen Softwareentwicklung vor.

Wie können darüber hinaus die Herausforderungen der Modellierung informatischer Systeme angegangen werden?

3.3.5 Konsequenzen

Die Modellierung von Informatiksystemen wird nicht nur erschwert durch Zielkonflikte, die durch die notwendige Formalisierung und das Spannungsverhältnis zwischen nicht-formaler Wirklichkeit und berechenbar-formalem System entstehen. Auch können diese Systeme durch ihr prozessierendes, interaktives Verhalten den Kontext, für

[7] Bekannt wurde Petri durch die so genannten Petri-Netze, mit denen System- bzw. Automatenzustände und -transitionen formal modelliert und als Graphen abgebildet werden können, siehe z.B. Priese und Wimmel (2008).

[8] Die Forderung, informatische Modellierung als Kommunikationsdisziplin zu betrachten, deutet auf den Wandel von Informatiksystemen vom Werkzeug zum Medium hin, wie z.B. von Schelhowe (1997) beschrieben.

und in dem sie modelliert wurden, verändern und zwar meist stärker, als dies statische Produkte tun können.

Interdisziplinarität

Eine formal-mathematische Perspektive und entsprechendes Vorgehen bei der Modellierung von Mensch-Computer-Interaktion vernachlässigt den Akteur Mensch und seinen unformalen, sozialen Kontext. Auf diesen einzugehen und diesen in die Modellierung einzubringen bedarf auch ‚unformaler' Herangehensweisen.

Weil sich Interaktion zwischen Computer und Menschen nur unzureichend formal modellieren lässt, hat sich die HCI in den vergangenen Jahrzehnten zu einer interdisziplinären, über Informatik hinausgehenden Fachrichtung entwickelt. Auf Bemühungen der HCI, durch interdisziplinäres Vorgehen die Kontexte der Benutzenden einzubeziehen und unformale Perspektiven auf Systemmodellierung zu finden, wird in Kapitel 4 eingegangen.

Modellierungshilfsmittel

Die Forderungen nach ‚bedachterer' und intensiverer Modellierung bei der Softwareentwicklung haben mittlerweile Beachtung gefunden, z.B. durch Einbeziehung der zukünftigen Nutzenden. Zahlreiche Methoden – so genannte Vorgehensmodelle – wurden dafür entwickelt. Hilfsmittel zur Softwareentwicklung wie z.B. CASE-Tools liefern Werkzeuge für den Modellierungsvorgang. Auch sind Entwicklungen wie die der Design-Rationale-Systeme zu beobachten, mit denen Entscheidungsgründe als pragmatische Merkmale dokumentiert werden sollen (siehe Abschnitt 5.2.2).

Gängige Software Engineering-Werkzeuge und -Sprachen wie UML bieten Notationsformen, um die Anforderungen an ein Informatiksystem, aber auch seine Architektur und sein Verhalten aus verschiedenen Perspektiven zu kommunizieren und zu definieren. Jedoch sind sie wenig intuitiv für Außenstehende. Außerdem beruhen diese Mittel auf formalen Sprachen und reichen daher nur bedingt aus, um soziale Kontexte zu berücksichtigen und das Verhalten von zukünftigen Nutzenden zu antizipieren.

Agile Softwareentwicklung

Ein dynamischer und informellerer Ansatz zur Systementwicklung wird mit agilen Methoden verfolgt. Aus Unzufriedenheit über Aspekte klassischer Software-Engineering-Methoden, wie z.B. deren Schwerfälligkeit, fassten im Jahre 2001 siebzehn Softwareentwickler und Methodenforscher in den USA ihre Ideen für eine ‚bessere' Vorgehensweise bei der Softwareentwicklung in einem „Agilen Manifest"[9] zusammen. Agile Methoden zur Implementierung und Projektsteuerung wie z.B. Extreme Programming (XP) (Beck, 2003) oder Scrum (Gloger, 2013) enthalten Elemente

[9] `http://agilemanifesto.org/iso/de/` (aufgerufen am 17.03.2011)

und Vorgehensmuster, die als Antwort auf die Modellierungs-Zielkonflikte angesehen werden können. Sie ermöglichen kleinschrittiges, iterativ-inkrementelles Vorgehen und eine engere Rückkopplung an Nutzende und Kund_innen im Entwicklungsprozess. Sie sind somit flexibler und lassen mehr Raum für Reflexion. Beispielsweise könnte das Einnehmen verschiedener Stakeholder-Rollen auf das System dazu beitragen, Intentionen verschiedener Modellsubjekte besser nachzuvollziehen. Ansätze, um die Subjekt-Objekt-Spaltung zu mindern, sind vermehrte Teamarbeit und mehr Feedback und Kommunikation mit Kunden und Nutzenden. Rückbezug und Wirksamwerden im sozialen Kontext können durch viele Iterationen in Verbindung mit Feedback und Kommunikation reflektiert werden. Dennoch muss wahrgenommene Wirklichkeit formalisiert werden und die grundlegenden Zielkonflikte werden auf die eine oder andere Weise das Produkt und den Modellierungsprozess beeinflussen.

3.4 Zusammenfassung und Schlussfolgerungen

Ziel dieses Kapitels war es, ein theoretisches Verständnis für den Prozess des ‚Machens' stofflich-digitaler Artefakte zu erarbeiten, um das Erstellen stofflich-digitaler Artefakte als auch die Artefakte, die währenddessen entstehen, theoretisch beschreiben und charakterisieren, aber auch verallgemeinern zu können. Dazu wurden stofflich-digitale Artefakte im Physical Computing verortet und als Tangible User Interfaces bezeichnet. Diese sind informatische Systeme.

Als eine Herangehensweise, um informatische Prozesse und insbesondere die Artefakte zu beschreiben, die dabei entstehen, wurde die Allgemeine Modelltheorie (AMT) von Stachowiak (1973) vorgestellt. Diese versteht Modelle als reduzierte Abbildungen eines Originals, welche zu einem bestimmten Zweck von einem Subjekt erstellt werden. Diese pragmatische Vorstellung betont den konstruktivistischen, subjektiven Aspekt jeglicher Modellbildung. Unterschieden wird zwischen graphischen, technischen und semantischen Modellen. Statt der Kategorie der semantischen Modelle wird in dieser Arbeit die Kategorie der textuellen Modelle nach Müller (1998) genutzt. Die AMT ist aufgrund ihrer Allgemeinheit nicht nur auf informatische Systeme, sondern auf alle Lebensbereiche und Artefakte anwendbar. Somit ist sie geeignet, den Eigenschaften selbstgemachter stofflich-digitaler Artefakte nachzugehen.

Da informatische Systeme mathematischen Ursprungs sind, d.h. formal beschrieben und definiert werden können, werden diese mittels konzeptueller Modellierung entwickelt. Konzeptuelle Modellierung ist zumeist gemeint, wenn im Informatikkontext das Wort Modellierung benutzt wird. Wie aufgezeigt wurde, kommen – bezogen auf die AMT – verschiedene Arten von Modellen in der Informatik vor, die über konzeptuelle Modelle hinaus gehen. Die Entwicklung informatischer Systeme kann als ein Modellieren von Modellen beschrieben werden. Es wird also eine Kette sehr verschiedener Modelle erstellt. Modelle werden in der Informatik erstellt, um Entwicklungsschritte

zu dokumentieren, zu kommunizieren, zu beweisen, zu testen. Sie können Prototypen sein, um Systemanforderungen zu überprüfen. Sie helfen auch dabei, Komplexität zu bewältigen, ein System besser zu verstehen und ein der Wirklichkeit entstammendes Modell in ein maschinell-ausführbares zu überführen.

Für viele Anwendungen, wie z. B. für Simulatoren, werden fachwissenschaftliche Modelle anderer Disziplinen in informatische, programmierbare Modelle überführt. Komplexer wird dieses Vorgehen im Kontext der HCI, wo kein eindeutiges, allgemeingültiges Modell für die Interaktion zwischen Mensch und Computer vorliegen kann. Zugleich werden Modelle informatischer Systeme von Subjekten zu bestimmten Zwecken konstruiert. So wurden Aspekte der Modellierung in der Informatik aufgezeigt, die durch den sozialen Kontext und die damit verbundene Diskrepanz zwischen formalem Zielmodell und nicht-formaler, zu modellierender Wirklichkeit entstehen und zu inhaltlichen Zielkonflikten führen. Dazu zählt das Wirksamwerden in voneinander getrennten, durch Formalisierung de- und rekontextualisierten sozialen Kontexten. Damit ist die Problematik der Subjektivität und der Intentionen der Modellierenden verbunden, die Teil der (oft unbewussten) pragmatischen Merkmale von Modellen sind. Gleichzeitig ist ein Digitales Medium zu einem Selbstbezug oder Rückbezug fähig, d.h. es vermag die Ausgangsbedingungen des in ihm realisierten Modells selbst zu verändern. Deshalb wurden mittlerweile Herangehensweisen entwickelt, die mehr Wert auf das Modellieren und Kommunizieren, aber auch das Iterieren der Modelle legen. Dabei sollen Stakeholder stärker und häufiger einbezogen werden und es soll flexibler auf sich ändernde Bedingungen reagiert werden. Beispiele dafür sind agile Methoden, die sich teilweise etabliert haben.

In Kapitel 2 wurde darauf hingewiesen, dass es wichtig für befähigendes Selbermachen sei, die Abstraktionen hinter Artefakten erkennen und auf das Artefakt beziehen zu können. Wie wir in diesem Kapitel gesehen haben, handelt es sich bei den Abstraktionen, die stofflich-digitalen Artefakten zugrunde liegen und deren Interaktionsmöglichkeiten festlegen, um von Subjekten formalisierte informatische Modelle. Sie basieren wiederum auf wahrgenommenen (z.B. gedanklichen) Modellen der Modellsubjekte, die einen Wirklichkeitsausschnitt beschreiben. Das Artefakt ist nie fertig, da die zugrundeliegenden formalen Anweisungen immer wieder durch anderes Interaktionsverhalten wirksam werden und von verschiedenen Menschen in verschiedenen Umgebungen unterschiedlich und immer wieder neu interpretiert werden. Um so wichtiger erscheint es, dass die abstrakten informatischen Modelle stofflich-digitaler Artefakte für deren Nutzende oder Nachbauende nachvollzogen und mit ihrem wahrgenommenen Verhalten in Verbindung gebracht werden können. So können auch die von den Modellierenden konstruierten Übersetzungen zwischen den Modellen eines Entwicklungsprozesses sichtbar werden. Das bedeutet im Falle von Dokumentations- oder Modellierungstools, dass sowohl die formalisierten Modelle als auch die wahr-

genommene Erscheinung des Artefakts und deren Zusammenhänge dargelegt werden sollten.

Stofflich-digitale Artefakte zu modellieren ist nicht nur Aufgabe der Informatik. Wie sich die Modellperspektive dieses Kapitels auf einen weiteren, interdisziplinären HCI-Kontext erweitern lässt, der alternative, unformalere Modellierung zulässt, soll Gegenstand des nächsten Kapitels sein.

Kapitel 4

Alternative Modellierungsansätze aus der Human-Computer Interaction

Das Entwickeln interaktiver informatischer Systeme ist nicht mehr nur Aufgabe der Informatik. Im Kontext der Human-Computer Interaction (HCI) sind auch Menschen mit nicht-informatischem Schwerpunkt, z.B. Designer_innen, an Entwicklungen beteiligt. Damit wird der Tatsache Rechnung getragen, dass informatische Modelle in sozialen Kontexten wirken, die sich nicht oder nur ausschnittsweise als formale, allgemeingültige Modelle fassen lassen.

Dieses Kapitel widmet sich alternativen, eher unformalen Ansätzen zur Modellierung informatischer interaktiver Systeme und konzentriert sich dabei insbesondere auf den Bereich Design bzw. Interaktionsdesign. Es wird angenommen, dass eine Designsicht hilft, Modellbildungsprozesse im Kontext des Making zu verstehen. Zum einen bestehen heutige Digitale Medien – z.B. als Tangible User Interfaces (TUIs) oder selbstgemachte stofflich-digitale Artefakte – nicht nur aus Software, die auf einem Standard-PC läuft. Sie haben einen stofflichen Köper, dessen materielle, händische Konstruktion eher ein kreatives unformales Vorgehen erfordert. Zum anderen kann diese Perspektive alternative Herangehensweisen an informatisches Modellieren aufzeigen, die genutzt werden können, um Amateur_innen ein verständliches Abbilden von Modellen beim Making zu ermöglichen.

Mit der Erweiterung des Modellierungsbegriffs um diese Perspektiven und dem Eruieren alternativer Modellierungsansätze nähert sich dieses Kapitel dem Selbermachen stofflich-digitaler Artefakte als Modellbildung im Amateurkontext weiter an. Zunächst wird die Entwicklung der HCI beschrieben, um zu zeigen, wie und warum Designgebiete Einzug gehalten haben. Daran anschließend werden Herangehensweisen an das Entwickeln bzw. Modellieren stofflich-digitaler Artefakte im Design und Inter-

aktionsdesign betrachtet. Dazu wird das in dieser Arbeit verfolgte Modellkonzept, das in den Anfängen der HCI aufkam und zwischenzeitlich in Vergessenheit geriet, auf diese Tätigkeiten und ihre Objekte übertragen. Auch werden für diese Arbeit interessante Perspektiven auf Modellbildung im Kontext der Mechatronik vorgestellt. Insbesondere für Entwicklungen von TUIs bzw. im Physical Computing sind Erkenntnisse aus diesem Gebiet interessant, weil ähnlich vielschichtige Systeme modelliert werden müssen.

4.1 Die Entwicklung der Human-Computer Interaction

> „In HCI we have witnessed the rise and fall of conceptual modeling in general. The 1980s focused on changing human behavior, which was captured in models to inform designs. Around 1990 a second wave of HCI questioned the usefulness of this type of approach, pointing out how human behavior is contingent and situated, and that human beings actively work around whatever technical solutions exist. In more recent years, this has been supplemented with a focus on emotion and experience [Verweis auf McCarthy und Wright (2004) – EK]. More than ever, this research points away from conceptual modeling.“ (Bodker, Mathiasen & Petersen, 2012, S. 54)

Dieses Zitat, das sich auf das Thema Usable Security (Bodker et al., 2012) bezieht, fasst Entwicklung und Wandel in der HCI und die schwindende Bedeutung konzeptuellen Modellierens treffend zusammen, wie im folgenden erläutert wird.

Aufgrund von Ereignissen wie der Softwarekrise und dem Einzughalten von Computern in Arbeitskontexten begann das Software Engineering in den 1970er Jahren, sich für Cognitive Engineering und Aspekte wie Usability zu interessieren, um informatische Systeme besser beherrschbar zu machen und Komplexität zu reduzieren (Carroll, 2013). In den 1980er Jahren wurden Computersysteme mit ausgeprägter Mensch-Maschine-Schnittstelle noch meist für den Arbeitsgebrauch entwickelt, also für eine relativ homogene Benutzergruppe in einem bestimmten Arbeitskontext mit eingrenzbarem Aufgabenfeld. Im privaten Bereich spielten sie keine nennenswerte Rolle. Im vorherigen Kapitel wurden Quellen aus den 1990er Jahren herangezogen, die anmahnten, beim Modellieren informatischer Systeme soziale Kontexte nicht außer Acht zu lassen und daher keine allzu formale, isolierte Modellbildung (wie z.B. in der Mathematik) zu betreiben. Allmählich verloren bei der Entwicklung solcher Systeme strikte, formale Ingenieurmethoden an Bedeutung: Die Anwendenden als menschliche Akteure, und nicht als bloße Faktoren (Bannon, 1991) rückten zunehmend in das Bewusstsein der HCI. Methoden für Participatory Design, verschiedene Techniken des Prototyping, Vorgehensweisen wie Contextual Inquiry und Methoden zur Usability-Erfassung entstanden.

Das Participatory Design oder auch Cooperative Design (Greenbaum & Kyng, 1991) entwickelte sich in Skandinavien als Antwort auf den Widerstand der Gewerkschaften. Diese waren mit der Rationalisierung von Arbeit durch den Einsatz von Computern

an Arbeitsplätzen nicht einverstanden und setzten sich gegen die Umgestaltung ihrer Arbeitsbedingungen (bis hin zum Wegfall einzelner Tätigkeiten wie z.B. der des Schriftsetzers) zur Wehr. Die in skandinavischen Ländern verbreiteten demokratischen Werte finden sich in der Idee des Participatory Design wieder, das den zukünftigen Nutzenden eine möglichst frühe und durchgehende Mitgestaltung und Teilhabe im Designprozess ermöglichen will (Bødker, Ehn, Sjögren & Sundblad, 2000).

Allmählich wurden die Interfaces kleiner und graphischer, Computersysteme hielten Einzug in private Bereiche des Lebens und dienen heute nicht mehr nur der Unterstützung von Arbeitsprozessen, sondern der alltäglichen Unterhaltung, Kommunikation und Assistenz in allen Bereichen des Lebens. Diese Entwicklung wird als Wandel oder zweite Welle (engl. „wave“) der HCI bezeichnet (vgl. Bødker, 2006; Sellen, Rogers, Harper & Rodden, 2009). Wie von Mark Weiser (1999) prognostiziert, sind Digitale Medien als informatische Systeme im Laufe der Zeit – und mit ihnen ihre Interfaces – unsichtbarer und kleiner geworden, so dass sie z.B. in Alltagsgegenständen verschwinden und zugleich allgegenwärtig geworden sind. Durch die zunehmende Vernetzung untereinander sind informatische Systeme heute als Digitale Medien zu Kommunikationsmedien geworden. Die Interaktion mit Digitalen Medien ist heute dynamischer – ihr Rückbezug vollzieht sich schneller. Durch Entwicklungen wie Ubiquitous Computing als Tangible User Interfaces oder mobile Geräte wachsen Software und Hardware teilweise in Form und Funktion zusammen (vgl. Buchenau & Suri, 2000), d.h. beide sind füreinander konzipiert und funktionieren nicht oder nur eingeschränkt auf anderen Systemen oder mit anderer Hardware.

Die Anwendungskontexte und -ziele heutiger Digitaler Medien verschwimmen zwischen Arbeit und privatem Gebrauch. Methoden, die sich auf eine effiziente Nutzbarkeit von Systemen konzentrieren, haben an Bedeutung verloren. Aspekten wie der User Experience und den Emotionen der Nutzenden wird vermehrt Beachtung geschenkt. Diese Entwicklung wird als dritte Welle („third wave“ (Bødker, 2006, S. 5)) in der HCI beschrieben. Wright und McCarthy (2008) beschreiben User Experience wie folgt:

> „[our approach, EK] suggests a different sensibility towards it, a different way of relating to familiar precepts such as know the user, iterative design, and user involvement. It requires us to see these familiar things in terms of felt life, empathy, and the aesthetics of everyday experience. User needs and requirements are not the focus of our enquiry. Rather the focus is an understanding of individuals, their concerns, desires, aspirations, values, and experiences. The relation between designer and 'user' is not an objective one in which the designer stands outside of the user's situation. Instead, it is one in which the designer and user are in mutually influencing, empathic dialog (...)“ (Wright & McCarthy, 2008, S. 19)

Experience Design kann beschrieben werden als Perspektive oder Haltung im Entwicklungsprozess und Verhältnis zu den Benutzenden des Systems. Wesentlich dabei ist sind Empathie und ein ganzheitlicher Blick auf die Benutzenden mit ihren Sin-

neserfahrungen, ihren Emotionen und ihren sozialen Kontexten, anstatt des alleinigen Beschränkens auf formale Modellierung. Erreicht werden soll dies auch durch interdisziplinäre Entwicklerteams (Wright & McCarthy, 2008).

Die Literatur, die im vorherigen Kapitel herangezogen wurde, um die Sichtweise auf Informatik als Modellierung zu beschreiben, stammt teilweise aus einer Phase der Informatik bzw. HCI, in der informatische Systeme weniger medial und weniger interaktiv waren. Die dort genannten Forderungen nach reflektiertem Vorgehen bei der Modellierung von Informatiksystemen aufgrund des Eingebundenseins informatischer Modelle in soziale Kontexte haben also in den vergangenen Jahrzehnten in der HCI zunehmende Beachtung gefunden. Zum einen geschieht dies, indem weitere Fachgebiete, insbesondere aus dem Designbereich, an Systementwicklungen beteiligt werden. Zum anderen sind neue Methoden für Softwareentwicklung entstanden, um den Charakter von Informatiksystemen als prozessierende und in einem interaktiven, sozialen Benutzungskontext befindliche Systeme zu berücksichtigen, anstatt nur eine Perspektive konzeptueller mathematisch-formaler Modellierung einzunehmen. So haben sich agile Methoden (siehe Abschnitt 3.3.5) etabliert, die in ihren Rahmenbedingungen weniger formal erscheinen und den Menschen mit verschiedene Sichtweisen und Interessen bewusst einbeziehen.

Um zukünftigen Nutzenden aktive Teilhabe und Gestaltungsmöglichkeiten schon während des Entwicklungsprozesses zu geben, wurden Methoden entwickelt, die auch für Nicht-Informatiker_innen und auch für Menschen, die sich sonst nicht mit der Gestaltung von interaktiven Systemen befassen, anwendbar sind (für einen Überblick siehe z.B. Sanders, Brandt & Binder, 2010). Solche Methoden oder Techniken sollen in dieser Arbeit als *unformal* (siehe auch Abschnitt 3.1.4) verstanden werden. Sie gehen oft über die Teilhabe an Entscheidungen hinaus, indem sie z.B. kreative Prozesse anregen und die Kommunikation der beteiligten Parteien ermöglichen oder fördern (z.B. mit Rollenspielen wie bei Macaulay, Jacucci, O'Neill, Kankaineen & Simpson, 2006). Neben partizipativen Methoden, die ursprünglich auf einen abgrenzbaren Arbeitskontext ausgerichtet waren mit eher homogenen Nutzergruppen, werden im Zuge des Experience Design neue Partizipationsmöglichkeiten diskutiert, die Nutzende mit ihrem gesamten Lebenskontext einbeziehen. Das Internet beispielsweise bietet neue Möglichkeiten des Feedbacks und der Interaktion mit Nutzenden und unmittelbaren Kontakt zu den Entwickelnden. Auch sollen Nutzende befähigt werden, Systeme selbst in einem gewissen Umfang an ihre eigenen Bedürfnisse anzupassen (Bødker, 2006; Ehn, 2008). Außerdem kann das Kooperieren mit Maker-Communitys als eine neue Form der Einbeziehung von Nutzenden angesehen werden, in der diese selbst als Entwickelnde tätig werden (siehe Abschnitt 2.2).

Es hat sich also in der HCI-Forschung und in der Praxis der Softwareentwicklung ein stärkeres Bewusstsein für den konfliktbehafteten Modellierungsprozess und die (sozialen) Kontexte, in denen die Modelle konstruiert und wirksam werden, durchge-

setzt. Der Benutzungskontext mit seinen nicht-formalisierbaren Aspekten wird ganzheitlicher betrachtet. Dazu beigetragen hat eine erweiterte Interdisziplinarität der HCI. Methoden und Ansätze aus anderen Fachgebieten wurden in der HCI adaptiert, an größeren Entwicklerteams sind Expert_innen aus mehreren Bereichen beteiligt, wie z.B. Pädagog_innen oder Designer_innen und vor allem Vertreter_innen der Zielgruppe des Systems. Mit der Einbeziehung von Designer_innen entwickelte sich die HCI auch zu einem Designgebiet, das nicht nur existierende Designgebiete aufnahm, sondern mit dem Interaktionsdesign (engl. *interaction design*) oder dem user experience design neue hervorgebracht hat (Carroll, 2013).

4.2 Interaktionsdesign

Im Jahre 1997 beschrieb Terry Winograd den Beginn einer neuen Profession mit einer anderen Herangehensweise an das Gestalten und Entwerfen von Systemen, als es die Informatik bis dahin leistete – das Interaktionsdesign.

> „In the midst of this interdisciplinary collision, we can see the beginnings of a new profession, which might be called 'interaction design'. While drawing from many of the older disciplines, it has a distinct set of concerns and methods. It draws on elements of graphic design, information design, and concepts of human-computer interaction as a basis for designing interaction with (and habitation within) computer-based systems. Although computers are at the center of interaction design, it is not a subfield of computer science.“ (Winograd, 1997, p. 157f.)

Er betont, dass es sich dabei *nicht* um einen Zweig der Informatik handelt, sondern um eine designartige Herangehensweise an das Medium Computer. Ähnlich wie Drucktechnologien Grafikdesign hervorgebracht haben oder das Entwerfen von Gebäuden nicht nur Aufgabe von Bauingenieur_innen sondern auch von Architekt_innen ist, so sah er auch mit der Technologie Computer und deren Vordringen in Lebensbereiche eine neue, andere Herangehensweise entstehen, die sich durch eine neue Perspektive auf Digitale Medien kennzeichnet. Eine Perspektive, die sich beim Entwerfen von Systemen weniger mit quantifizierbaren Faktoren und Bedingungen befasst, sondern mit nicht-formalisierbaren menschlichen Dingen wie Wertesystemen und Bedürfnissen. Da sich bisherige Designbereiche wie z.B. Grafikdesign nur unzureichend auf das neue, interaktive und virtuelle Medium anwenden ließen, würde sich ein neues Designfeld entwickeln (Winograd, 1997).

Anderthalb Jahrzehnte später sind Winograds Prognosen Wirklichkeit geworden. Löwgren (2013) ordnet interaction design in der „Encyclopedia of Human-Computer Interaction“ als Designprozess ein. Zu den wesentlichen Eigenschaften, die Interaktionsdesign als Design kennzeichnen, zählt Löwgren das Ändern aktueller Situationen durch das Entwerfen neuartiger Artefakte, das Visionieren von Zukunft und das Aufzeigen von Lösungen, die nicht (nur) einer Analyse der existierenden Proble-

me entspringen. Dabei wird mit externen Repräsentationen gearbeitet, um Ideen zu reflektieren. Neben funktionalen Aspekten werden auch technische, ästhetische und ethische Dimensionen adressiert, die sich gegenseitig beeinflussen bzw. in einem Spannungsfeld zueinander stehen (Löwgren, 2013).

Die externen Repräsentationen können – soviel sei hier vorweg genommen – im Sinne der AMT als Modelle beschrieben werden. Deren Bedeutung im (Interaktions-) Design ist daher für diese Arbeit von besonderem Interesse und wird im Laufe dieses Kapitels aufgegriffen und ausgeführt.

In der Literatur finden sich unterschiedliche Beschreibungen von Interaktionsdesign. In dieser Arbeit soll Interaktionsdesign wie in den oben angeführten Beschreibungen von Winograd (1997) und Löwgren (2013) verstanden werden, die dessen Designeigenschaften betonen.[1] Zunächst soll genauer betrachtet werden, was unter ‚dem' Design und ‚designen' zu verstehen ist, um ein allgemeineres Verständnis von designassoziiertem Vorgehen zu entwickeln und daraus die in Kapitel 3 eingeführte Perspektive der informatischen Modellbildung zu erweitern sowie alternative Modellierungsansätze zu identifizieren.

4.3 Was kennzeichnet Design?

Der Begriff des Designs wird im Englischen vielfach gebraucht und meint als Verb „‚etwas aushecken', ‚vortäuschen', ‚entwerfen', ‚skizzieren', ‚gestalten', ‚strategisch verfahren'" (Flusser, 1993, S. 9), als Substantiv „‚Vorhaben', ‚Plan', ‚Absicht', ‚Ziel', ‚böswilliger Anschlag', ‚Verschwörung', ‚Gestalt', ‚Grundstruktur'" (Flusser, 1993, S. 9). Design schließt im Englischen damit auch den mentalen Prozess, das sich ausdenken von irgendetwas, mit ein. Hier jedoch soll Design als Bezeichnung eines Fachgebiets und als Tätigkeit von Menschen mit entsprechender Berufs(aus)bildung verstanden werden, die etwas – z.B. ein stoffliches oder virtuelles Objekt und damit verbundene menschliche Interaktion – gewissenhaft gestalten und formen.

Doch was ist dieses Design? Eine Definition wagt kaum eine_r, und oft wird Design umschrieben, indem Tätigkeiten, Herangehen und Abgrenzung zu anderen Gebieten vorgenommen werden. Auch das „Design Dictionary" von Erlhoff und Marshall (2008) verzichtet auf eine Definition, sondern führt verschiedene, teils historisch begründete Beschreibungen und Aspekte an. Der Designtheoretiker Wolfgang Jonas (1999) charakterisiert Design wiederum mit den Adjektiven „nutzen-orientiert" (sic), „illustrativ", „antizipativ", „generativ", „integrativ" und „kontext-sensitiv" (Jonas, 1999) (deutsche Übersetzungen anhand Jonas, 2004).

[1] Auch Hochschulen für Interaktionsdesign, so wie das Copenhagen Institute of Interaction Design, folgen einem solchen Verständnis. Auf deren Website heißt es beispielsweise: „Interaction Design is a practice that combines traditional design disciplines with socio-technological trends. It leads to intuitive solutions for products, software and services." `http://ciid.dk/about/` (aufgerufen am 17.03.2014)

Die Notwendigkeit für Design entsteht aus technologischen Entwicklungen (z.B. industrielle Produktion, Druck- und Bautechnologien, Computer), die in menschliche Lebenswelten vordringen und so ein entsprechend darauf ausgerichtetes Gestalten – Designen – künstlicher Artefakte erforderlich machen. Anders als frühere (Kunst-) Handwerker_innen produzieren Designende ihre Produkte in der Regel nicht selbst (Erlhoff & Marshall, 2008). Im Gegensatz z.B. zur Informatik, die sich im Kern mit Computern beschäftigt, oder bestimmten Wissenschaften, die sich auf ein Betätigungsfeld konzentrieren, ist es laut Erlhoff und Marshall nicht möglich, dem Design solch ein Tätigkeitsfeld zuzuordnen:

> „This description [referring to seeing design as process and product, EK] implies that design does not exist as an exclusive (...) discipline but rather acts to integrate a range of academic, economic, environmental, scientific and artistic insights, knowledge, and opinions together with the everyday process of lived experience into the artifacts, systems, and processes of our constructed lives." (Erlhoff & Marshall, 2008, S. 108)

Design beschäftigt sich mit sozialen, menschlichen Prozessen und muss sich deshalb immer mit mehreren Gebieten beschäftigen.

> „Design, by its very nature, is involved in all the social processes and when these are ignored the outcome will be compromised and unsatisfactory and, thus, it has to be grounded in high quality research across many fields." (Erlhoff & Marshall, 2008, S. 108)

Um Lösungen für Probleme zu finden, muss ein_e Designer_in lediglich „just enough" (Erlhoff & Marshall, 2008, S. 108) der an einem Projekt beteiligten Gebiete und Perspektiven kennen. Design ist damit inter- oder transdisziplinär.

Aufgrund der Schwierigkeit einer Definition soll Design auch hier als ein bestimmtes Herangehen und als eine Haltung oder Perspektive aufgefasst werden. Um dies zu erläutern, sollen im Folgenden Konzepte, mit denen Design oft beschrieben wird, herangezogen werden: Design als reflexive Praxis und die Kontrastierung der Herangehensweisen im Design mit denen der (Natur-)Wissenschaften und des Ingenieurwesens.

4.3.1 Reflection in Action

Design folgt keinem linearen Verlauf von der Idee zur Spezifikation und Umsetzung, sondern kann als ein Hin und Her betrachtet werden zwischen Ideen, Prototypen und weiteren Repräsentationen, in dessen Prozess auch Ideen aus intuitiven Erwägungen komplett verworfen werden und wieder neue entstehen können (Löwgren & Stolterman, 2004).

Um zu beschreiben, wie Designer_innen vorgehen und ihr Handeln implizit reflektieren, gilt Donald A. Schöns Konzept der „reflection-in-action" (Schön, 1983, S. 49) als einflussreich und wird in der Literatur oft herangezogen. In „The Reflective Practitioner: How Professionals Think in Action" beschreibt er seine Beobachtungen, wie

professionelle Praktiker, z.B. Designer_innen, implizites Wissen anwenden und erwerben, indem sie ihre professionellen Fähigkeiten unmittelbar während ihrer Tätigkeit aufbauen und anpassen. Schön kritisiert mit seinem Ansatz die geringere Wertschätzung praxisbasierten Wissens gegenüber wissenschaftlicher Rationalität (Mareis, 2011).

Als reflection-in-action bezeichnet Schön, wie die Praktizierenden, während sie in ihre Tätigkeit vertieft sind, ihr momentanes Handeln reflektieren und in der Situation unmittelbar darauf reagieren. Sie passen ihr Handeln durch fortlaufendes Experimentieren an, wenn die gewohnten Lösungsansätze sie nicht weiterbringen. Er erläutert dies am Beispiel eines improvisierenden Jazzmusikers. Reflection-in-action findet als eine Konversation mit der Situation statt. Der Handelnde handelt oft spontan und möglicherweise unbewusst, ohne innezuhalten und nachzudenken. Dieses implizite Wissen – im Falle von Designer_innen ihr Designwissen – ist nur im praktischen Handeln abrufbar, Schön bezeichnet es als „knowing-in-action" (Schön, 1983, S. 49f.). Die Tätigkeit des Entwerfens beschreibt er als „Reflective conversation with the situation" (Schön, 1983, S. 76) bzw. dem Material in der Situation (Schön, 1992). Er nennt es „seeing–moving–seeing" (Schön, 1992, S. 134). *Moving* könnte z.B. das Zeichnen beim Skizzieren sein, und *seeing* nicht nur *sehen*, sondern auch *erkennen*. Als charakteristisch für Designer_innen nennt er implizites, körperlich-sensorisches Wissen:

> „A designer's knowing-in-action involves sensory, bodily knowing. The designer designs not only with the mind but with the body and senses (...)." (Schön, 1992, S. 133)

Insgesamt zeigt das Konzept von Donald Schön die gleichberechtigte Rolle impliziten Handelns und praxisbasierten Wissens in kreativen Prozessen auf. Er zeigt damit, dass auch solches Wissen und Vorgehen eine Berechtigung einnimmt und sich praxisorientierte Tätigkeiten nicht allein auf explizites theoretisches Wissen und analytisches Handeln berufen können. Und es zeigt die Berechtigung von Wissen, das aus solchen Situationen hervorgeht. Solches Wissen und Handeln soll als grundlegend für Designprozesse angesehen werden, deren Charakter im weiteren Verlauf näher betrachtet und abgegrenzt wird.

4.3.2 Design, Engineering und Naturwissenschaften

Um Design zu beschreiben, wird es oft mit den Naturwissenschaften, aber auch mit dem Ingenieurwesen kontrastiert.[2]

Der Designer und Designtheoretiker Otl Aicher (1991) unterscheidet Designer von Künstlern und Ingenieuren: Ein Ingenieur verfahre nach logischen Sprüngen, messe und zähle, rechne und folge Kausalitäten und denke dabei linear. Ein Maler wiederum leite ästhetische Qualitäten aus seiner Zielvorstellung ab, ihm gehe es um eine Aussage. Der Designer hingegen säße zwischen den Stühlen (vgl. Aicher, 1991, S. 67f.):

[2] Wobei manche Designtheoretiker ‚engineering' als Designtätigkeit miteinschließen, was sicher auch dem weiter gefassten Begriff des Wortes ‚to design' im Englischen geschuldet ist.

> „der designer kann sich weder auf eine rational analytische arbeitsmethode zurückziehen, die alles in quantitäten auflöst und quantifizierbar macht, noch kann er sich darauf beschränken, qualitäten zu erzeugen, ordnungen der anschauung, der farbe, der form.“ (Aicher, 1991)

Die Arbeitsmethode des Designers sei komplexer und würde über die Tätigkeit des Ingenieurs und Künstlers hinaus gehen, indem es seine Aufgabe sei, als eine Art Moralist zu werten (Aicher, 1991).[3]

Jonas (1999) unterscheidet Designdenken von wissenschaftlichem, ingenieurtypischem und künstlerischem Denken:

> „design thinking is different from scientific thinking (analytic, reductionist, aiming at explanation), it is different from engineering thinking (aiming at efficient functionality), and it is different from artistic thinking (taking the artist's self as primary criterion). For all these reasons design thinking has to claim theoretical and methodological autonomy.“ (Jonas, 1999)

Nigel Cross (2001) verweist auf den Gegensatz von Design als erfindend und konstruktiv zu den Naturwissenschaften als problemlösend und analytisch. Lawson (1979) untersuchte das Herangehen an Probleme von Studierenden der Naturwissenschaften (engl. „science“) und der Architekturgestaltung (engl. „architectural design“) aus Abschlussjahrgängen, sowie jeweils auch mit Studierenden aus Anfangsjahrgängen. Die angehenden Naturwissenschaftler_innen gingen Probleme analytischer an, wohingegen die Designstudierenden eher verschiedene Lösungen ausprobierten, um eine passende zu finden. Das Vorgehen der Naturwissenschaftsstudierenden beschreibt er als problemfokussiert, das der angehenden Designer_innen als lösungsorientiert (Lawson, 1979).

Diese Beschreibungen und Unterscheidungen zwischen dem (idealisierten) Vorgehen von Ingenieur_innen und Designer_innen muss aber nicht auf als solche Ausgebildete beschränkt sein, sondern wurde auch an anderer Stelle beobachtet als unterschiedliches Problemlöseverhalten. Turkle und Papert (1991) beobachten bei jungen Programmieranfänger_innen unterschiedliches Vorgehen und charakterisieren diese als „planners“ und „bricoleurs“ (Turkle & Papert, 1991, S. 169). Sie unterscheiden zum einen einen planenden Stil, der ingenieurartig, strukturiert vorgeht. Dazu gehört ein schrittweises ‚top-down‘ Verfeinern. Am anderen Ende gibt es einen Bricolage-Stil, der von spielerischem Aushandeln mit dem Material bestimmt ist, vergleichbar mit dem Vorgehen eines Künstlers oder Kochs, der sich durch die verfügbaren Zutaten inspirieren lässt, ähnlich Schöns Beschreibung von der reflexiven Konversation mit dem Material (Turkle & Papert, 1991).

[3] Diese Einstellung bringt Mareis (2011) mit der politischen demokratischen Orientierung Aichers (die auch an der HfG Ulm verfolgt wurde), in Verbindung.

4.3.3 Interaktionsdesign als Design

Was kann aus diesen Ausführungen für das Interaktionsdesign gefolgert werden?

Interaktionsdesign ist auch interdisziplinär, und eine wesentliche beteiligte Disziplin ist die Informatik. Dennoch ist Interaktionsdesign von Tätigkeiten wie Software oder Hardware Engineering durch seine andere Herangehensweise an das Entwerfen und Formen von Computersystemen abzugrenzen. Die Tätigkeit ist weniger analytisch und problemorientiert mittels Formalisierung und Quantifizierung, sondern lösungsorientiert, konstruktiv und generativ. Wie oben erwähnt wurde, müssen Interaktionsdesignende ‚nur genug' der jeweils relevanten Fachgebiete und Anwendungsbereiche wissen und gleichzeitig Designmethoden kennen, mit denen sie zu einer Lösung kommen können. Dabei stehen insbesondere menschliche Aspekte und Nutzungskontext im Fokus des Entwickelns einer Lösung, und nicht (nur) technische Möglichkeiten. Durch diese schwer zu fassenden Faktoren ist das Vorgehen von intuitiven, im Handeln reflektierten Entscheidungen geprägt, anstatt aus der analytischen Betrachtung eines Problems formal eine Lösung abzuleiten.

Diese Beschreibungen sollen nicht ausschließen, dass ein und dieselbe Person ein System designt und implementiert und dabei mal generativ und lösungsorientiert mit einem Designblick auf die Aufgabe arbeitet, und mal analytisch aus Ingenieur- bzw. Informatiksicht handelt und dementsprechend auch auf verschiedene Art und Weise modelliert. Der Begriff Design wird in dieser Arbeit benutzt, um die Möglichkeit ‚unformaler' alternativer Herangehensweisen an das Entwerfen informatischer Systeme aufzuzeigen. Sie soll nicht missverstanden werden als eine strikte Kategorisierung von Tätigkeiten mit der Bezeichnung ‚Design' bzw. von Menschen, die diese ausüben. Wie sich designassoziiertes Vorgehen als Modellierungstätigkeit beschreiben lässt, wird im Folgenden nachgegangen.

4.4 Modelle in Design und Interaktionsdesign

> „entwerfen heißt, modelle zu konstruieren." (Aicher, 1991, S. 195)

Wie dieses Zitat des Designers und Designtheoretikers Otl Aicher in seinem Buch „die welt als entwurf: schriften zum design" (1991) über die entwerfende Tätigkeit von Designer_innen verrät, ist Modellieren auch im Design eine zentrale Tätigkeit. Dabei meint Aicher (1991) den Begriff des Modells hier – ähnlich wie die AMT – als Mittel zur Erkenntnisgewinnung, Zugang zur Wirklichkeit und als Begriffskonstruktion (vgl. auch Gänshirt, 2011, S. 151).

4.4.1 Design als Modellbildung

Das Vorgehen bei Designtätigkeiten wurde als ein reflektierendes Arbeiten mit externen Repräsentationen beschrieben. In Abschnitt 3.2 wurde informatische konzeptuelle

Modellierung thematisiert, die anhand logischer Folgerungen vorgehen kann. Auch wurde Einschränkungen in Form von Zielkonflikten beim Entwickeln von informatischen Systemen für Menschen aufgezeigt.

Betrachtet man das typische Vorgehen von Designer_innen wie es oben beschrieben wurde, so handelt es sich nicht um analytische, kausale problem-orientiert vorgehende Modellbildungsprozesse, sondern um kreative, generative konstruktive Modellierung. Wenngleich Softwareingenieur_innen ihre Probleme prinzipiell auch mittels formal-analytischer konzeptueller Modellierung lösen können (wenngleich dieses dann nicht unbedingt zufriedenstellend für die Nutzenden ist, siehe Abschnitt 3.3), so sind Interaktionsdesignende auf andere Herangehensweisen angewiesen, da ihr Hauptaugenmerk auf den Nutzenden und den für sie zufriedenstellenden Lösungen liegt.

4.4.2 Modelle, Mock-ups, Prototypen

Auch Designer_innen arbeiten mit Modellen. Jedoch wird der Begriff des Modells sehr eng gefasst, ähnlich wie Informatiker_innen damit eher 3D-Modelle und Objekt-orientierte Modellierung assoziieren mögen. Im „Design Dictionary“ definiert die Designtheoretikerin und Designerin Melanie Kurz (2008) den Begriff des Modells – abweichend vom obigen Zitat Aichers – aus Designperspektive als Skizze oder dreidimensionales (u.U. virtuelles) Objekt:

> „Models are used in virtually every aspect of the design process: in the visualization (...) of form, the development of function, the communication of processes, the evaluation of alternatives, and so on. They come in a wide variety of forms: two-dimensional sketches, three-dimensional objects (scale model, functional model) and virtual representations (3-D computer rendering).“ (Kurz, 2008, S. 262)

Modelle bezeichnen im Designkontext i.d.R. konkrete, materialisierte Entwürfe, wie etwa ein Architekturmodell zur dreidimensionalen Darstellung eines Gebäudes in verkleinertem Maßstab. Modelle werden im Designkontext oft in einer Kategorie neben Skizzen, Prototypen und Mock-ups genannt (z.B. in Koskinen, Zimmerman, Binder, Redstrom & Wensveen, 2011). Bei Mock-ups handelt es sich um Low-Fidelity-Prototypen mit niedriger Wiedergabetreue (Brandt, 2007), also wenigen Attributen des Originals. Im weiteren Verlauf dieses Kapitels wird der Begriff des Modells jedoch nicht nur als Modell im Sinne des Designs, sondern allgemeiner im Sinne der AMT (siehe Abschnitt 3.1) verstanden. Damit wird Modell auch synonym zu dem, was Designende als Skizzen, Prototypen oder Mock-ups bezeichnen, verwendet. Doch wie lassen sich solche Modelle in die Modellkategorien der AMT einordnen?

Dreidimensionale materielle statische Modelle sind nach der AMT in der Unterkategorie statisch-mechanische Modelle der physikotechnischen Modelle einzuordnen. Ein digitales CAD-Modell oder eine Skizze sind als graphische Modelle einzuordnen und fallen dort in vollschematische Abbildungen als Untergruppe ikonischer Bildmodelle (vgl. Thomas, 2002, S. 60, Stachowiak, 1973, S. 168 und Abbildung 3.1 auf Seite

37). Wohingegen, wie in Abschnitt 3.2.4 aufgezeigt wurde, ein Diagramm im Software Engineering zu den symbolischen Darstellungsmodellen gezählt werden kann, das sich auf einer höheren Ebene der semantischen Modelle befindet. Generell sind Modelle von Designer_innen oft ikonischer Natur, und somit eher allgemeinverständlich als formalsprachliche Informatikmodelle.[4]

4.4.3 Funktion von Modellen beim Interaktionsdesign

Design kennt, wie oben angeführt wurde, unzählige Anwendungsfelder wie z.B. Architektur, Industriedesign oder Modedesign. In dieser Arbeit werden vorwiegend Quellen herangezogen, die sich mit Interaktionsdesign beschäftigen, um die Bedeutung von Modellen im Designkontext zu eruieren.

Auch der Designprozess kann als Modellbildungsprozess bezeichnet werden. Er ist gekennzeichnet durch ein Iterieren zwischen konkreten, meist materiellen Modellen verschiedener Granularität. Modelle als externe Repräsentationen haben dabei weitere Funktionen, die über das Kommunizieren von Ideen hinaus gehen.

In Abschnitt 3.2.4 wurde die Rolle von Prototypen im Softwareengineering beschrieben. Diese sind vornehmlich auf das Erheben und Überprüfen von Systemanforderungen ausgerichtet. Wie im (Software) Engineering dienen Modelle im Design der Kommunikation und Dokumentation von Ideen und Entwürfen und unterstützen auch die Zusammenarbeit. Allerdings haben sie noch weitere Funktionen, die von der Art und Weise, wie Designer_innen arbeiten, herrühren. Neben ihrer veranschaulichenden und überprüfenden Funktion (wie im Software Engineering), haben Prototypen im Designkontext auch eine generative Funktion. Prototypen – und damit konkrete Modelle – sind im Design kreatives Material und Reflexionsobjekte (Löwgren & Stolterman, 2004). Sie ermöglichen es Designenden ihr Handeln zu reflektieren und den Entwurfsraum[5] zu erforschen. Prototypen im Design sind weniger dazu da, Lösungen zu überprüfen und zu bestätigen, sondern stellen ein Mittel der Designenden dar, Probleme der als flexibel bezeichneten Designaktivitäten zu entdecken und zu identifizieren (Lim et al., 2008).

Modelle zur Reflexion – ‚models' to think with

Anfassbare Mock-ups oder Prototypen haben also eine wichtige Rolle im Designprozess als „things-to-think with" (Brandt, 2007, S. 179), die die Entwerfenden zur Reflexion anregen. Designende können ihre Ideen aus einer neuen Perspektive betrachten und möglicherweise bisher unbedachte Möglichkeiten oder Hindernisse entdecken (Löwgren & Stolterman, 2004). Die Objekte werden Teil und Mittel der „reflection-in-action" (s.o.) und helfen so auch, implizites Wissen explizit zu machen. Prototypen

[4] Kulturell bedingte Sehgewohnheiten vorausgesetzt.

[5] Englisch „design space" (Lim, Stolterman & Tenenberg, 2008, S. 7:2)

werden damit – ähnlich wie im Lernverständnis des Konstruktionismus (siehe Abschnitt 2.5.3) – zu „objects to think with“ (Papert, 1980, S. 150). Gedanken und Auffassungen werden durch Prototypen externalisiert. Modelle haben also eine wichtige Funktion zur Reflexion im Designprozess, indem sie Konzeptionen und Ideen von außen wahrnehmbar machen.

Modelle zur Kommunikation und als Bindeglied

Brandt (2007) weist auch auf die Rolle von Mock-Ups als verbindende Objekte in partizipativen Designprozessen hin. In Designprozessen dient das Erstellen konkreter Modelle nicht nur der Verbindung von Iterationsstufen, sondern auch als vermittelndes Medium zwischen beteiligten Parteien. Diese Funktion von Modellen wird in der Soziologie als „boundary objects“ (Star & Griesemer, 1989) bezeichnet. Als Begründer dieser Theorie gelten Star und Griesemer (1989), die im Rahmen einer ethnografischen Fallstudie an einem Museum beobachteten, wie Akteur_innen aus verschiedenen Communities-of-Practice ihre verschiedenen Sichtweisen und daraus resultierenden Spannungen anhand solcher Objekte lösten. Anstatt die unterschiedlichen Anliegen der einzelnen Akteure zu kanalisieren, findet bei den boundary objects ein „many-to-many mapping“ (Star & Griesemer, 1989, S. 390) statt, indem verbindliche Übergangspunkte ausgehandelt werden durch die verschiedenen beteiligten Parteien. Um zwischen mehreren Sichtweisen zu vermitteln und zu einem gemeinsamen Verständnis zu gelangen, ist demnach die Entwicklung von boundary objects zentral. Sie können sowohl konkret als auch abstrakt sein, z.B. in Form materieller Repräsentationen, aber auch als Methoden vorkommen (Star & Griesemer, 1989).

Boundary objects vereinen verschiedene, sich überschneidende Welten und erfüllen deren jeweilige Informationsbedürfnisse. Sie erlauben es, eine gemeinsame Identität der beteiligten Parteien herzustellen und aufrechtzuerhalten. Sie verfügen zwar über eine allgemein-erkennbare Struktur, durch die sie sich als Übersetzungsmedium eignen, haben aber verschiedene Bedeutungen in den verschiedenen sozialen Lebenswelten. Boundary objects fungieren somit als temporäre Anker sowie als Bindeglieder, durch die nicht einfach die Sichtweise einer Subjektwelt den anderen Akteur_innen aufgezwungen wird (Star & Griesemer, 1989).

Das Konzept der boundary objects wird nicht nur im Zusammenhang mit HCI und (Interaktions-)Design (z.B. bei Brandt, 2007), sondern auch in Engineering-Prozessen (z.B. als „intermediary objects“ bei Boujut & Blanco, 2003) aufgegriffen, um Entwicklungsprozesse, an denen mehrere Stakeholder beteiligt sind, zu beschreiben. Es zeigt die verbindende, die Kommunikation und soziale Prozesse unterstützende Funktion von Objekten, die im Sinne der AMT als Modelle bezeichnet werden können. Voraussetzung ist jedoch, dass solche Modelle für alle Beteiligten verständlich genug sind und an ihre jeweiligen Lebenswelten anknüpfen. Das Konzept verweist auch darauf,

dass ein und dasselbe Modell verschiedene Bedeutungen für die jeweiligen Beteiligten haben kann.

4.4.4 Modellbildungskonflikte im Designkontext

Auch für Designaufgaben erstellte Modelle haben pragmatische Merkmale, die diese kennzeichnen, aber nicht immer offensichtlich sind. Da Entscheidungen oft intuitiv und weniger rational getroffen werden, sind diese mitunter für Außenstehende nicht nachvollziehbar. Dementsprechend kann es auch hier zu Zielkonflikten zwischen den Absichten von Designenden oder Entwickelnden und der Interpretation der Nutzenden kommen, die sich als Kommunikationsprobleme äußern.

Crilly, Maier und Clarkson (2008) beschreiben die Beziehung zwischen der Intention von Designer_innen und der Experience der Konsument_innen anhand eines integrierten Kommunikationsmodells. Dazu erweitern sie u.a. das Kommunikationsmodell von Don Norman (1990). Diesem zu Folge kommunizieren Nutzende und Entwickelnde eines Systems nur über das Abbild dessen, z.B. das stoffliche Erscheinen, die Funktionsweise, wie es reagiert, Bedienungsanleitungen und andere Hinweise, mit denen das Produkt oder System ausgeliefert wird. Die Nutzenden erklären sich die Funktionsweise des Systems über diese Abbilder. Entwickelnde und Nutzende haben daher i.d.R. nicht dieselben mentalen Modelle in Bezug auf das System, weshalb die Entwicklenden dafür sorgen müssten, dass das Produkt oder System in jeder Hinsicht konsistent mit dem konzeptuellen Modell der Anwendenden ist und seine Funktionsweise durch sein Erscheinen selbst erklärt (Norman, 1990).

Crilly et al. (2008) berücksichtigen nun das Verhalten von Designer_in und Konsument_in in ihren jeweiligen sozialen und kulturellen Kontexten und ihren dadurch bedingten Erwartungen und Motivationen, die ihre Experience mit dem Produkt konstituieren und beeinflussen. Ihr Kommunikationsmodell erweitert das z.T. dekontextualisierte, einfache Modell Normans im Sinne des Experience Design zu einem zusammenfassenden Konzept. Das Artefakt steht dabei als Kommunikationsmedium im Mittelpunkt des theoretischen Modells. Dieses Kommunikationsmodell zeichnet sich dadurch aus, dass es verschiedene Kommunikationswege zwischen Artefakt, Konsument_innen, Designenden und Herstellenden abbildet und dabei die jeweiligen Subjektperspektiven und deren Intentionen und Kontexte als rückwirkend miteinbezieht.

Crilly et al. (2008) sprechen von Artefakten und unterscheiden dabei nicht zwischen digitalen und nicht-digitalen Produkten. Das Artefakt als Modell nimmt die Rolle eines Mediums zwischen den Beteiligten ein, über das diese kommunizieren. Generell verhält es sich beim Design von Artefakten also ähnlich wie bei der informatischen Modellierung – Konflikte bezüglich der Modellsubjekte und ihrer getrennten Kontexte entstehen. Auch beim Interaktionsdesign werden komplexe informatische Systeme modelliert – wenn auch in frühen Phasen der Entwicklung mit unformalen Herangehensweisen einer Designdisziplin anstatt mittels formaler konzeptueller Modellierung.

Jedoch wird dabei ein System entwickelt, das spätestens bei der technischen Implementierung formaler Modellierung bedarf und daher konfliktbehaftet ist (siehe Abschnitt 3.3).

Hugh Dubberly beleuchtet in seinem Artikel „Models of Models“ (Dubberly, 2009) Rolle und Funktion des Modellierens im Interaktionsdesign. Modellieren ist demnach wichtig, um sich der eigenen Konzeption, d.h. des mentalen Modells des zu entwickelnden Systems bewusst zu werden, und dieses ggf. anzupassen und zu erweitern. Insbesondere durch das Externalisieren mentaler Modelle und der damit verbundenen Intentionen im Designprozess können die Beteiligten ihre Konzeptionen überprüfen und zu einem gemeinsamen Modell, d.h. Verständnis, kommen. Wobei sich auch durch den Prozess des Externalisierens das mentale Modell bereits verändern kann (Dubberly, 2009).

> „Models help bridge the gap between observing and making—especially when systems are involved (as in designing for interaction, service, and evolution).“ (Einleitung von H. Dubberly zu Dubberly & Evenson, 2011, S. 75)

Modelle können also Interaktionsdesignenden helfen, die potentiellen Konflikte im Designprozess bzw. das Verhältnis von Designenden und Nutzenden des Systems und die Rolle des Systems dazwischen (bzw. die zugrunde liegenden Intentionen und Interpretationen) besser zu verstehen und zu antizipieren. Aufgrund der Komplexität von interaktiven Systemen müssen vielerlei Aspekte berücksichtigt werden, was nur geschehen kann, wenn mehrere Modelle mit verschiedenen Sichtweisen oder Granularitäten erstellt werden, so dass alle Aspekte des Systems berücksichtigt werden können (Dubberly, 2009).

Auch aus Perspektive des Interaktionsdesigns ist es daher wichtig, pragmatische Merkmale wie Intentionen sowohl gegenüber Nutzenden als auch gegenüber anderen Parteien, die an der Entwicklung eines Systems beteiligt sind, zu kommunizieren. Dies bedeutet zum einen, Intentionen mitzuteilen, die zu bestimmten Entscheidungen und damit Modellen geführt haben, aber auch, sich durch konkrete Modelle der eigenen Auffassungen bewusst zu werden und diese mitteilen zu können. Insbesondere beim Entwickeln komplexer Systeme scheint ein Abbilden verschiedener Aspekte eines Systems in Modellen nicht nur für unmittelbar an der technischen Implementierung Beteiligte, sondern auch für Interaktionsdesignende notwendig.

Digitales Material

Beim Interaktionsdesign – aber auch zunehmend in anderen Designgebieten[6] – werden Modelle mit Hilfe digitaler Werkzeuge erstellt. Doch was bedeutet dies für die Modelle?

Digitale Prototypen wie auch digitale Artefakte als Zielmodelle bringen Besonderheiten mit sich hinsichtlich ihrer Materialität, was bei der Wahl geeigneter Prototypen zu Herausforderungen führt. Löwgren und Stolterman (2004) bezeichnen digitales

[6] Zum Beispiel beim Produktdesign mit CAD-Software.

Material deshalb als Material ohne natürliche Eigenschaften (engl. „quality“). Digitales Material bietet unendlich viele Möglichkeiten, Dinge zu erstellen, zu visualisieren, und durch technische Neuentwicklungen kommen ständig neue hinzu.[7] Durch die nicht vorhandenen materiellen Eigenschaften fehlen Form und Funktion bestimmende Begrenzungen. So werden digitale Tools zu einem besonderen Material für Prototyping und erfordern eine bedachte Wahl beim Konzipieren von Prototypen (Lim et al., 2008). Im Kontext der AMT bedeutet dies, dass Attribute und Reduktionsgrad bedachter gewählt werden sollten als wenn mit Material gearbeitet wird, welches von Natur aus eine bestimmte Reduktion impliziert (z.B. ein Papierprototyp).[8]

Digitale Materialien bringen beim Modellieren auch Eigenschaften mit sich, die Formalisierungsschritte erforderlich machen. Auch wenn diese von einem digitalen Tool vollzogen werden, so beeinflussen sie möglicherweise den Prototyp. Also ist schon beim Erstellen von digitalen Modellen zu bedenken, dass bereits auf dieser Ebene Zielkonflikte entstehen können als auch, dass Designentscheidungen beim Entwickeln von Modellierungstools sich auf die später damit zu erstellenden Modelle auswirken.

4.4.5 Ergänzende Bemerkungen zu Interaktionsdesign als Modellbildung

Interaktionsdesign wird in dieser Arbeit als ein designartiges Herangehen an die Entwicklung informatischer Systeme betrachtet, das das technische Implementieren nicht miteinschließt. Demzufolge kann und will sich Interaktionsdesign bei der Modellierung nicht auf logische formale Herleitung berufen, wie es beim Software-Engineering möglich ist. Ziel ist es nicht, die technisch ausgefeilteste Lösung zu finden, sondern die menschlich beste. Das Modellieren kann vielmehr als konstruktives, gestaltendes nicht-lineares Modellieren beschrieben werden. Konkrete, anfassbare, gegenständliche Modelle spielen eine wichtige Rolle und dienen der Reflexion der Designideen, die mitunter in einer Auseinandersetzung mit dem Modellierungsmaterial entstehen.

Die in diesem Kapitel eingenommene idealisierte Sichtweise soll nicht missverstanden werden, als dass Designtätigkeiten nur die beschriebenen designtypischen Herangehensweisen zulassen würden. Auch dort kann mit Abstraktionen, Formalisierungen und Methoden gearbeitet werden, die theoretisches Wissen voraussetzen.[9] Die Designperspektive wurde gewählt, um – zumindest für frühe Entwicklungsphasen – einen alternativen, mit gegenständlich-konkreten Modellen assoziierten Zugang aufzuzeigen, der mit analytisch-abstrahierendem, informatischem Vorgehen kontrastiert wird. Da-

[7] Aktuell z.B. einfach zugängliche digitale Fabrikationstechnologien.

[8] Allerdings kann die Notwendigkeit zur Formalisierung durchaus Beschränkungen mit sich bringen. Beispiel sei eine 3D-Applikation, mit der ein 3D-Objekt zwar skaliert werden kann (eine Multiplikation der Werte als eine formale, mathematische Operation), das aber keine Möglichkeit bietet, mit der Hand das Objekt zu formen. Oder beim Design eines Interfaces, wo das Tool nur einen begrenzten Satz an Schriftarten bereit hält. Diese sind um so entscheidender, wenn digitale Pläne wieder in materielle Objekte umgewandelt werden, z.B. mittels digitaler Fabrikation (siehe Abschnitt 2.4.1).

[9] Man denke etwa an die Arbeit von Architekt_innen.

mit soll gezeigt werden, dass auch konkretere und damit ggf. allgemeinverständlichere Modelle und Herangehensweisen während informatischer Modellierungsprozesse zulässig sind, die teilweise andere Eigenschaften als informatische Modelle besitzen.

4.5 Modellierungsansätze aus dem Mechatronikkontext

Das Modellieren stofflich-digitaler Artefakte wurde bislang als ein Modellieren beschrieben, das in den Gebieten Informatik und Design verankert ist. Zu Beginn von Abschnitt 3 wurden stofflich-digitale Artefakte mit Tangible User Interfaces (TUIs) in Verbindung gebracht. Die Gestaltung und Entwicklung von TUIs ist nicht nur Aufgabe von Interaktionsdesign und Informatik, sondern bringt weitere Disziplinen wie Materialwissenschaften, Robotik und Mechatronik mit ein (Hornecker, 2011). Hier soll nun ein Blick auf die Mechatronik geworfen werden, die Mechanik, Elektronik und Informatik verbindet, und die auch mit Arduino-Technologien assoziiert wird (Jung, Martelaro, Hoster & Nass, 2014).

4.5.1 Parallele Modellketten mit verschiedenen Perspektiven

Zunächst soll eine Sicht auf Mechatronikentwicklungsprozesse als Modellbildungsprozess von Buur und Andreasen (1989)[10] herangezogen werden, die von einem ähnlichen Modellverständnis wie die vorliegende Arbeit ausgeht.[11]

Die Entwicklung eines Mechatronikprodukts beschreiben Buur und Andreasen als „a combination of mechanical, electronic and software engineering“ (Buur & Andreasen, 1989, S. 155). Mechatronikprodukte können mit TUIs oder auch mit von Maker_innen entwickelten Artefakten verglichen werden, die ebenfalls eine Mischung aus Hardware-, Software- und Mechanikentwicklung enthalten.

Buur und Andreasen (1989) beschreiben Beispiele von Entwurfsmodellen in der Mechatronik, die im Laufe eines Projekts entwickelt werden. In einer Grafik stellen sie die Bereiche Software, Elektronik und Mechanik als drei parallele Spalten dar (siehe Abbildung 4.1). In jedem dieser Bereiche wird eine eigene, von den anderen Bereichen

[10] Noch erwähnenswert sei hier, dass Jacob Buur sich mittlerweile mit TUIs befasst, siehe z.B. Hornecker und Buur (2006). Des Weiteren beschäftigt er sich mit Nutzerbeteiligung und Entwicklung innovativer partizipativer Methoden, d.h. mit dem Eingebettetsein von Systemen in sozialen Kontexten und den damit verbundenen Problematiken, und entwickelt Lösungen dafür (siehe `http://www.sdu.dk/en/om_sdu/institutter_centre/c_spire/people/jacob`, aufgerufen am 13.01.2013).

[11] Sie beschreiben die Eigenschaften von Modellen als „*object*“, „*modelled properties*“, „*purpose*“ und „*user*“ (Buur & Andreasen, 1989, S. 157f.). Object meint das Zielprodukt bzw. die Vorstellung dessen und entspricht in etwa dem Original in der AMT (siehe Abschnitt 3.1). Properties stehen für die Funktion oder das Aussehen eines Modells und damit für die gewählten Attribute und können mit dem Abbildungs- und Verkürzungsmerkmal der AMT verglichen werden. Purpose entspricht dem Zweck eines Modells und user der Zielgruppe des Modells. Beim Erstellen eines Modells wählen die Modellierenden aufgrund von purpose und user den „*code*“ (Sprache, Symbole etc.) und das „*medium*“ (z.B. Zeichnung, dreidimensional-materiell) (Buur & Andreasen, 1989, S. 157f.).

isolierte Modellfolge modelliert. Für die Software entstehen Diagramme, schriftliche Beschreibungen, Softwareprototypen und schließlich Programmcode. Im Bereich Elektronik wird aus dem ursprünglichen Blockdiagramm u.a. ein auf einem Breadboard aufgesteckter Versuchsaufbau, die schließlich in ein PCB-Layout münden. Für die Mechanik werden erste Skizzen u.a. in funktionale stoffliche Modelle übertragen und schließlich als detaillierte technische Zeichnungen abgebildet. Es entstehen isolierte parallele Modellketten (vertikal verknüpfte Modelle in Abbildung 4.1), die (zunächst) unabhängig von den Modellen der jeweils anderen Fachbereiche entwickelt werden. Das Beispiel zeigt zum einen, dass das Erstellen der Modelle nicht nur in Design- und

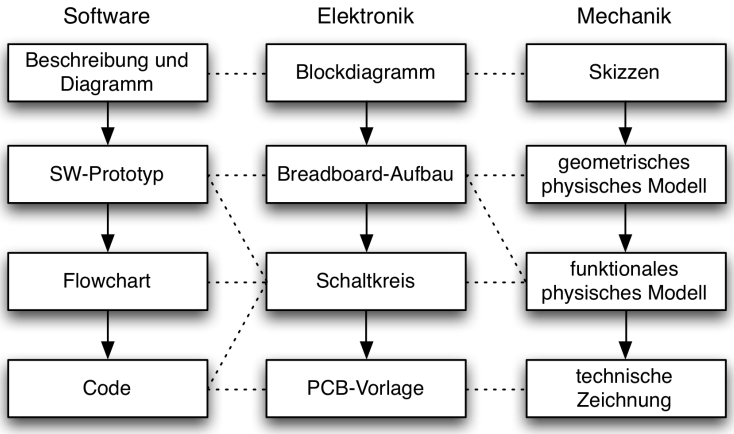


Abbildung 4.1: Mögliche Modellketten im Modellierungsprozess eines Mechatronikprodukts (in Anlehnung an Grafik in Buur und Andreasen, 1989, S. 157) mit geforderten beispielhaften Verkettungen zwischen Fachgebieten (gepunktete Linien)

Informatikprojekten als ein Bilden von Modellketten (siehe Abschnitt 3.2.2) im Laufe eines Entwicklungsprozesses beschrieben werden kann, sondern auch in Elektronik und Mechanik bzw. Mechatronik. Zum anderen wird deutlich, welche Herausforderungen die Modellierung eines komplexen Projekts mit sich bringt, wenn dessen Bestandteile nicht nur in einem Fachgebiet zu verorten sind (z.B. Software) und es diverse Modelle, Komponenten und Fertigungsschritte beinhaltet, die wieder zusammengefügt werden müssen. Zu solchen Projekten gehören generell auch stofflich-digitale Artefakte.

Entsprechend bemängeln Buur und Andreasen (1989) die isolierte Modellierung in den Teildisziplinen und fordern neue Entwurfsmodelle, die alle Modelle der beteiligten Teildisziplinen in einer frühen Phase zusammen bringen und die das Produkt als Ganzes sehen, um mögliche Risiken (v.a. hinsichtlich der Kosten) früh erkennen zu können. Daher kann gefolgert werden, dass es bei einem Modellierungsprozess eines technischen Systems wie z.B. einem stofflich-digitalen Artefakt, das Modellketten verschiedener Disziplinen enthält, wichtig ist, dass die verschiedenen Modelle frühzeitig über die Disziplinen hinweg miteinander in Verbindung gebracht werden. Es sind

nicht nur Modellketten relevant, die chronologisch mit zunehmender Detailgetreue für ein Teilsystem erstellt werden. Auch die parallel in einzelnen Disziplinen entstehenden Modellstränge sollten miteinander in Bezug gebracht werden, so dass wir gar von *Modellnetzen* sprechen können (eingezeichnet als gepunktete Linien in Abbildung 4.1). Verbindungen zwischen Modelle sollten also – bezogen auf Abbildung 4.1 – nicht nur vertikal, sondern auch horizontal hergestellt werden können.

4.5.2 Übergänge zwischen konkreten und abstrakten Modellen

Ein weiterer Ansatz zu Modellierung im Mechatronikkontext ist für diese Arbeit relevant, der sich mit verschiedenen Abstraktionsstufen von Modellen solcher Systeme auseinander setzt und diese für Lernzwecke nutzt. Die Entwicklungen der Bremer artec-Gruppe[12] zeigen, wie im Mechatronikkontext für abstrakte Konzepte konkrete Zugängen beim Modellieren technischer Systeme genutzt werden können (Hornecker, Robben & Bruns, 2001; Schäfer & Bruns, 2001; Müller, 1998). Benutzende können mit Modellierungstools für Fabrikplanung oder für Pneumatik im Berufsschulunterricht Modelle aus konkreten stofflichen Bausteinen konstruieren, die automatisch in virtuelle Repräsentationen (z.B. Diagramme) übersetzt werden (siehe Systembeschreibungen in Abschnitt 5.3.2). Zugrunde liegen diesen Systemen die Konzepte der Übergänge und der Erfahrungsorientierung.

Das „Konzept der Übergänge“ (Hornecker et al., 2001, S. 208) sieht vor, dass konkrete materielle Modellen mit ihren abstrakten Repräsentationen im System verknüpft sind, so dass verschiedene Benutzungsschnittstellen zur Verfügung stehen. So ist den Modellierenden ein ständiger Wechsel zwischen gegenständlichen konkreten Modellen und abstrakten virtuellen Repräsentationen (und ggf. weiteren Zwischenstufen) möglich. Die abstrakten Modelle hinter den konkret erfahrbaren Modellen werden offengelegt und ein experimenteller Umgang mit diesen komplexen Systemen wird ermöglicht (Hornecker et al., 2001).

Das Konzept der Erfahrungsorientierung sieht vor, dass die Tools an den Vorerfahrungen der Nutzenden anknüpfen, indem diese die für sie bekannte oder gewohnte Modellrepräsentation benutzen können, um weitere Abstraktionsstufen zu explorieren. Damit können solche Modellierungstools wiederum das Machen und Integrieren neuer Erfahrungen fördern (Müller, 1998).

Der vorgestellte Ansatz konzentriert sich auf das Verbinden und Übersetzen konkreter und abstrakter Modelle, um sie z.B. Berufsanfänger_innen näher zu bringen. Mittels konkreter Objekte und durch das Verknüpfen konkreter mit abstrakten Repräsentationen können komplexe Systeme, so wie z.B. TUIs, für Amateur_innen verständlicher dargestellt werden. Der Ansatz berücksichtigt dabei die Zusammenhänge zwischen Modellrepräsentationen, die sich in Abbildung 4.1 vorwiegend in derselben vertikalen Spalte befinden. Somit lassen sich die Vorteile konkreter ‚Design‘-Objekte

[12] `http://www.arteclab.uni-bremen.de` (aufgerufen am 05.06.2014)

wie Unterstützung der Kommunikation durch verbindende Objekte, (intuitives) Einbringen von Erfahrungen und damit verbundener Reflexion am Material auch beim Modellieren und Explorieren komplexer technischer Systeme zu Nutze machen.

4.5.3 Einsichten aus der Mechatronik-Perspektive

Design wurde in diesem Kapitel als Herangehensweise an entwerfende Tätigkeiten beschrieben, bei denen Gestaltende sich nicht auf rationale Entscheidungen berufen können, um bestimmte Qualitäten ihres Produkts zu erzielen. Das bedeutet nicht, dass dieses Herangehen nur bei Designtätigkeiten angewandt werden kann. Vielmehr, so wie bei Turkle und Papert (1991) und Schön (1983) oben beschrieben, handelt es sich hier um eine Herangehensweise, die in allen Berufszweigen und Aktivitäten oder Entscheidungsfindungen vorkommen kann. Wie die hier beschriebenen Beispiele zeigen, können auch Ingenieurtätigkeiten zumindest während Phasen des Prototyping von intuitivem Erfahrungswissen profitieren und kreativ in Konversation mit konkreten, stofflichen Objekten Ideen entwickeln (siehe auch Brereton & McGarry, 2000).

Dieser Exkurs zeigt, dass Modellierung in der Informatik und im Interaktionsdesign nur zwei von vielen Perspektiven darstellen, um Modellierung beim Entwickeln von stofflich-digitalen Artefakten zu beschreiben. Gleichzeitig zeigt die herangezogene Literatur, dass das Konzept der Modellierung nach AMT nicht nur auf Design und Informatik anwendbar ist, sondern auch auf Entwicklungstätigkeiten in anderen HCI-Bereichen. Somit ist die Betrachtung der involvierten Tätigkeiten als Modellbildung ein geeigneter Ansatz, um das Entwickeln stofflich-digitaler Artefakte theoretisch zu beschreiben. In Bezug auf die Entwicklung von Modellierungstools für interdisziplinäre Projekte kann gefolgert werden, dass entsprechende Tools verschiedene Perspektiven und Modellarten frühzeitig über Fachgebiete hinweg integrieren sollten und dass das Einbeziehen verschieden konkreter und abstrakter Modellrepräsentationen und deren Verknüpfungen beim Verstehen eines Systems helfen können.

4.6 Zusammenfassung und Schlussfolgerungen

Dieses Kapitel hatte zum Ziel, alternative Modellierungsansätze jenseits von konzeptueller formal-informatischer Modellierung im Kontext der HCI zu ergründen und damit das in dieser Arbeit verfolgte Modellkonzept, das in den Anfängen der HCI aufkam und zwischenzeitlich in Vergessenheit geriet, zeitgemäß auf den Kontext des Interaktionsdesigns zu übertragen und auszuweiten. Damit wurde auch ‚unformales' Herangehen als Modellierung beschrieben, um sich dem Selbermachen stofflich-digitaler Artefakte als Modellbildung anzunähern.

Es wurde aufgezeigt, dass das Entwerfen und Entwickeln informatischer Systeme nicht nur Aufgabe der Informatik ist. Da Informatiksysteme in soziale Kontexte eingebettet sind und mittlerweile nicht mehr nur der Arbeitserleichterung dienen, sondern

in alle Lebensbereiche vorgedrungen sind, hat sich die HCI im Laufe ihrer Entwicklung auch anderen Fachgebieten wie Sozialwissenschaften und Design zugewandt, um Systeme zu entwickeln, die die Benutzung als User Experience begreifen. So ist mit dem Interaktionsdesign ein Designgebiet Bestandteil von Entwicklungsprozessen informatischer Systeme geworden, dem wesentliche Aspekte von Designhandeln zugesprochen werden können.

Weil designtypisches Vorgehen eine Perspektive auf alternative, ‚unformalere' Modellbildung informatischer Systeme zumindest für Entwurfsphasen eröffnet, beschäftigte sich dieses Kapitel vorwiegend damit. Auch bestehen TUIs, wie auch selbstgemachte stofflich-digitale Artefakte, nicht nur aus einem Computer, sondern auch aus einem stofflichen Köper, der zumindest beim Erstellen von Prototypen ein kreatives, designartiges und unformales Modellieren erlaubt.

Bei Design wie auch Interaktionsdesign handelt es sich um interdisziplinäre Tätigkeiten, deren Hauptaugenmerk auf den sozialen, menschlichen Aspekten liegt, für die etwas gestaltet wird. Design wurde weniger als Fachgebiet, sondern als Herangehensweise und Perspektive beschrieben. Die Designenden müssen sich mit den beteiligten Fachgebieten und Aspekten nicht im Detail auskennen, sondern gerade genug wissen, um ihre Designentscheidungen zu fällen. Das typische Vorgehen im Design wurde kontrastiert zu Ingenieurwesen und Naturwissenschaften als weniger analytisch und problem-orientiert, sondern als konstruktiv und lösungsorientiert. Dabei sind nichtlineare Prozesse üblich, in denen mit externen, oft gegenständlichen Repräsentationen als Reflexionsobjekte wie in einer Konversation mit dem Material gearbeitet wird. Dadurch ist auch das Interaktionsdesign in der Lage, menschliche Aspekte zu berücksichtigen, denn es verfügt über Voraussetzungen und Herangehensweisen, die soziale Kontexte und menschliche Werte und Bedürfnisse einbeziehen. So kann die Beteiligung von Designer_innen am Entwicklungsprozess helfen, den sozialen Kontext informatischer Modelle zu berücksichtigen.

Modelle spielen auch beim designassoziierten Vorgehen eine wesentliche Rolle. Der Modellierungsprozess ist gekennzeichnet durch das Iterieren zwischen verschiedenartigen konkreten Modellen, die die Designidee repräsentieren. Für diese Arbeit wurde der in Designkreisen übliche Modellbegriff weiter gefasst und auch Modelle, die als Prototypen, Mock-ups oder Skizzen bezeichnet werden, einbezogen. Nach AMT-Kategorien sind typische Modelle im Design als ikonische Bildmodelle und physikotechnische, statisch-materielle Modelle einzuordnen. Die Funktion dieser Modelle geht jedoch über das Kommunizieren und Testen oder Verifizieren von Entwürfen hinaus. Wir haben deshalb Modelle zur Reflexion als ‚models-to-think-with' – angelehnt an Konzepte im Design und im Konstruktionismus (siehe Abschnitt 2.5.3) – und als verbindend entsprechend des Konzepts der boundary objects betrachtet. Im Gegensatz zur formal-mathematischen Modellierung, die Teil informatischer Entwicklungsprozesse ist, kann Modellierung im Interaktionsdesign (das aus der Sichtweise dieser

Arbeit nicht die technische Implementierung beinhaltet) als ein konstruktives, nichtlineares und gestaltendes Modellieren charakterisiert werden, das sich nicht in erster Linie aus logischer mathematischer Kausalität bedingt. Modellieren dient nicht dazu, ein Problem wie eine mathematische Aufgabe zu lösen, sondern den Designraum zu explorieren und dabei verschiedene Ideen zu verfolgen und zu entwickeln, aber auch zu verwerfen. Konkrete, meist anfassbare, gegenständliche Modelle spielen dabei eine wesentliche Rolle.

Beim designtypischen Vorgehen können sich Zielkonflikte als Kommunikationsproblem zwischen Designer_innen und Konsument_innen eines Produkts bemerkbar machen. Ein Artefakt enthält pragmatische Merkmale wie Intentionen der Designenden, die von den Nutzenden interpretiert werden müssen. Auch zwischen Designer_innen werden Intentionen nicht immer ausreichend kommuniziert. Beim Gestalten interaktiver Systeme kommen durch die Notwendigkeit der Formalisierung informatische Zielkonflikte hinzu, spätestens wenn das von Designer_innen erstellte Modell für die Implementierung formalisiert werden muss bzw. für die Formalisierung notwendige Informationen den Softwareentwickler_innen kommuniziert werden müssen. Gleichzeitig arbeiten Interaktionsdesigner_innen auch mit digitalem Material wie z.B. Designtools, die Formalisierung notwendig machen – oder diese scheinbar übernehmen und die Formalisierungmechanismen nicht erkennen lassen.

Zuletzt zeigte ein Exkurs in den Kontext der Mechatronik, die durch ihre Nähe zu TUIs für die HCI relevant geworden ist, dass das Konzept von Entwicklungsprozessen stofflich-digitaler Artefakte als Modellbildungsprozesse über verschiedene Fachgebiete hinweg anwendbar ist. Buur und Andreasen (1989) setzen sich mit den verschiedenen Facetten auseinander, die auch das Modellieren eines Tangible User Interfaces mit sich bringt, und die entstehen, wenn verschiedene Disziplinen aus verschiedenen Sichtweisen zusammen arbeiten. Demnach sollten verschiedene aufeinanderfolgende Modelle einzelner Aspekte, die z.B. die Software oder die Produktgestaltung repräsentieren, während des Modellierungsprozesses nicht isoliert voneinander betrachtet, sondern zueinander in Beziehung gebracht werden. Wir haben diese in Anlehnung an die Modellketten aus Abschnitt 3.2.4 als Modellnetze bezeichnet. Als ein weiterer für diese Arbeit relevanter Ansatz wurden Konzepte der Übergänge und der Erfahrungsorientierung herangezogen. Diese zeigen, dass es für das Verständnis solcher komplexen Systeme für Laien hilfreich sein kann, wenn abstrakte und konkrete Repräsentationen derselben Systemsichten miteinander in Bezug gebracht und wechselseitig verändert werden können. Der konkrete Zugang erlaubt den Modellierenden, an vorhandene Erfahrungen anzuknüpfen. Der Ansatz ist ein Beispiel dafür, wie der designartige Umgang mit konkreten Modellen genutzt werden kann, um auch abstrakte Modelle konkreter Systeme Amateur_innen näher zu bringen. Im Kontext des Selbermachens stofflich-digitaler Artefakte könnte das Darstellen der Zusammenhänge von konkreter

und abstrakter Repräsentation für Amateur_innen nützlich sein, wenn sie ihre Artefakte für sich selbst oder für andere verständlich dokumentieren wollen.

Das Designen von TUIs und das Selbermachen stofflich-digitaler Artefakte beinhalten auch das Entwerfen stofflicher Objekte, die nicht nur aus einem programmierten Computer bestehen, sondern auch eine Form und Gestalt haben, die es zu modellieren gilt. Diese kann insbesondere beim Selbermachen auch ohne Formalisierungsschritte geschaffen werden.[13] Somit enthält das Erstellen stofflich-digitaler Artefakte auch Modellierungsaktivitäten, die schon immer ein designassoziiertes, kreatives Herangehen erlauben. Die erarbeitete Perspektive auf Designobjekte als Modelle im Sinne der AMT kann dabei helfen, auch Abbildungsmöglichkeiten für diese Arten von Modellen zu finden.

Auch bei einer designbasierten Entwicklungstätigkeit handelt es sich um eine professionelle Herangehensweise, die nicht immer intuitiv für Amateur_innen sein kann. Demzufolge können Designmethoden nicht unbedacht für selbermachende Amateur_innen übernommen werden. Wie und mit welchen Mitteln Tools mit eher unformalen Zugängen Laien beim Modellieren und Abbilden ihrer Modelle stofflich-digitaler Artefakte unterstützen wollen, wird im nächsten Kapitel – dem fünften – betrachtet.

Durch dieses Kapitel wurde die Vorstellung vom Erstellen informatischer Systeme als Modellierung aus Abschnitt 3.2 auf einen (Interaktions-)Designkontext übertragen. Dabei zeigte sich, dass auch ikonische konkrete Modelle und intuitives Vorgehen eine Berechtigung für Modellierungsprozesse informatischer Systeme haben. In Kapitel 6 wird diese Designperspektive mit der Informatikperspektive auf Modellierung in Verhältnis gesetzt und auf das Selbermachen stofflich-digitaler Artefakte übertragen.

[13] Im professionellen Bereich gilt dies eher während Phasen des Prototypings. Um ein Produkt schließlich produzieren zu können, müssen auch hier irgendwann ‚formalere' Modelle wie z.B. technische Zeichnungen erstellt werden.

Kapitel 5

Methoden und Tools für Modellbildungsprozesse

In den vorangegangenen Kapiteln wurde das Selbermachen stofflich-digitaler Artefakte im Makerkontext verortet und mit dem Prinzip der Modellbildung aus Informatik- und Designperspektiven theoretisch in Verbindung gebracht. Bisher haben wir u.a. festgestellt, dass

- Maker_innen narrative How-tos zur Dokumentation ihrer Projekte benutzen,
- es wichtig ist, auch abstrakte Eigenschaften nachvollziehbar und in Bezug zu ihrem konkreten Erscheinen abzubilden, um die Realisierung und damit ein Verständnis Digitaler Medien zu ermöglichen,
- das Erstellen stofflich-digitaler Artefakte sowohl aus Informatik- als auch aus Designsicht als Modellierung und die dabei entstehenden Objekte als Modelle beschrieben werden können,
- Modelle auch pragmatische Merkmale besitzen, die mitzuteilen notwendig ist, um sie nachvollziehen zu können,
- es auch unformales, mit Design assoziiertes Herangehen an die Modellierung informatischer Systeme geben kann und dieses sich beim Vermitteln abstrakter Sachverhalte durch entsprechende Übergänge zu Nutze gemacht werden kann,
- es sich bei Modellbildungsprozessen stofflich-digitaler Artefakte um ein Bilden von Modellketten oder gar -netzen handelt.

Wie bereits besprochen wurde, ist das Modellieren stofflich-digitaler Artefakte eine komplexe Tätigkeit, in deren Folge unzählige Modelle verschiedener Ausschnitte und verschiedener Perspektiven entstehen können. Aufgrund des Kontexts und der Ziele dieser Arbeit sind folgende Modellierungsprozesse relevant:

- Modellieren als das fortlaufende Dokumentieren der während eines Projekts entstehenden Modelle des vollständigen selbstgemachten Artefakts, so dass dies nachvollzogen werden kann.
- Modellieren als das Übertragen einer unformalen Projektidee in formale, vom Computer ausführbare abstrakte Modelle. Dies umfasst das eher unformale, allgemeinverständliche Abbilden der angedachten Interaktionsgestaltung als auch die Formalisierung – meist Programmierung – dieser Modelle.

Offen bleibt bislang, wie Werkzeuge mit eher unformalen Zugängen Laien beim entsprechenden Modellieren und Dokumentieren ihrer stofflich-digitalen Artefakte unterstützen können. Deshalb widmet sich dieses Kapitel Tools und Methoden im Maker- und HCI-Kontext, die Erkenntnisse diesbezüglich liefern können. Darunter befinden sich auch die Programmierumgebungen Amici und MoTo sowie das Dokumentationstool VirtualLab, an deren Entwicklung und Erprobung die Autorin beteiligt war. Weil diese Tools Einfluss auf die vorliegende Arbeit hatten und zum Teil im Laufe der Arbeit für weitere Untersuchungen und Weiterentwicklungen herangezogen werden, werden sie ausführlicher als andere Tools behandelt. Anschließend werden Empfehlungen von Wissenschaftler_innen für Tools für Makeraktivitäten betrachtet.

5.1 Dokumentieren von Projekten im Makerkontext

Die im Makerkontext am weitesten verbreiteten, öffentlich sichtbaren Abbildungen von Modellierungsprozessen sind Webplattformen für die so genannten How-tos, die bereits in Abschnitt 2.5.1 besprochen wurden. Daneben werden Makerprojekte im Internet auch in verkürzter Form, d.h. konzentriert auf das Endergebnis ohne ausführliche Anleitung, präsentiert. Für beide Formen von Projektdokumentationen stehen diverse Webplattformen und Applikationen zur Verfügung, von denen einige hier exemplarisch vorgestellt werden.

5.1.1 Chronologische Anleitungen

How-tos bilden in chronologischer Reihenfolge den Modellierungsprozess ab. Dabei beschreiben sie auch unterschiedliche Teilaspekte des Artefakts (z.B. als Skizzen und Programmcode). Sie werden von Selbermachenden benutzt, um Projektabläufe eher unformal darzustellen. Ihr Zweck ist neben dem Dokumentieren des eigenen Vorgehens, als Vorlage und Anleitung für andere zu dienen.

Die Plattform *Instructables*[1] kann als Maker-Community bezeichnet werden. Dort können Selbermachende ihre Projekte als How-tos – Schritt-für-Schritt-Anleitungen mit Fotos, Beschreibungen, Videos und weiteren Anhängen – einstellen. Auf Fotos

[1] `http://www.instructables.com` (aufgerufen am 24.02.2013)

können bestimmte Bereiche mit Tags versehen werden. Andere Community-Mitglieder können das ‚instructable' kommentieren und bewerten.

HowDo[2] ist eine Smartphone-Applikation, mit der Schritt-für-Schritt-Anleitungen aus Fotos oder als Video direkt vom Smartphone hochgeladen werden können. HowDo soll dem Austausch von Anleitungen dienen. Anstatt erklärende Worte als Text hinzuzufügen, können die Fotos oder Videos mit Sprachaufnahmen versehen werden. Somit ermöglicht diese App eine unkomplizierte, direkte Einbindung von Projekten mittels Smartphone und integriert sich vermutlich besser in den Ablauf von Projekten, weil kein zusätzlicher Transfer von Fotos und kein Formulieren von Texten notwendig sind. Für Projekte, die digitale Materialien enthalten, z.B. 3D-Modelle oder digitale Programme und Pläne für Physical Computing, eignet sich diese App weniger, da solche Dateien nicht unmittelbar eingebunden werden können.

5.1.2 Dokumentationen fertiger Artefakte

Weitere Plattformen begnügen sich mit verkürzten Projektdarstellungen, die oft nur das fertige Artefakt zeigen.

Ein Beispiel ist die Plattform *Diy.org*[3], die sich an Kinder richtet und diese zum Selbermachen animieren will. DIY-Projekte können dokumentiert werden mittels Foto(s) und Beschreibung. Ziel scheint vor allem das Zeigen und Teilen eigener (fertiger) Projekte zu sein.

Auch das *VirtualLab*[4] verfolgt diesen Ansatz. Das VirtualLab ist im Zusammenhang mit TechKreativ-Workshops und dem Projekt EduWear (Reichel, Osterloh, Katterfeldt, Butler & Schelhowe, 2008) entstanden. Es soll als Projektplattform dienen, in der Teilnehmer_innen der Workshops ihre Projekte austauschen. Zielgruppe sind Kinder und Jugendliche aus TechKreativ-Workshops. Die erste Version des VirtualLab wurde von 2009 bis 2011 graphisch, strukturell und technisch überarbeitet. Im VirtualLab können registrierte Nutzer_innen Projekte einstellen mit dem Namen des Projekts, einem Foto, einer Beschreibung (Text), dem Programmcode als Dateianhang und einer Liste verwendeter Bauteile und Komponenten, die per drag-and-drop aus einem virtuellen EduWear-Koffer eingefügt werden können. Teammitglieder des Projekts können genannt werden und ob das Projekt in einem TechKreativ-Workshop entstanden ist. Projekte können kommentiert werden. Auf der Startseite wird eine Auswahl von Projekten angezeigt, die nach beliebt, neu oder zufällig sortiert werden können.

Im VirtualLab sowie bei Diy.org werden Projekte als Ergebnis und nicht als Prozess abgebildet. Projektdokumentationen können als Modelle betrachtet werden, die fertige Artefakte abbilden. Welche Merkmale und Eigenschaften mit dem VirtualLab erstellte Modelle enthalten, wird in Abschnitt 7.3 dieser Arbeit analysiert.

[2] `http://www.how.do` (aufgerufen am 20.03.2013)
[3] `http://www.diy.org` (aufgerufen am 20.03.2013)
[4] `http://www.dimeb.de/drupal` (aufgerufen am 20.03.2013)

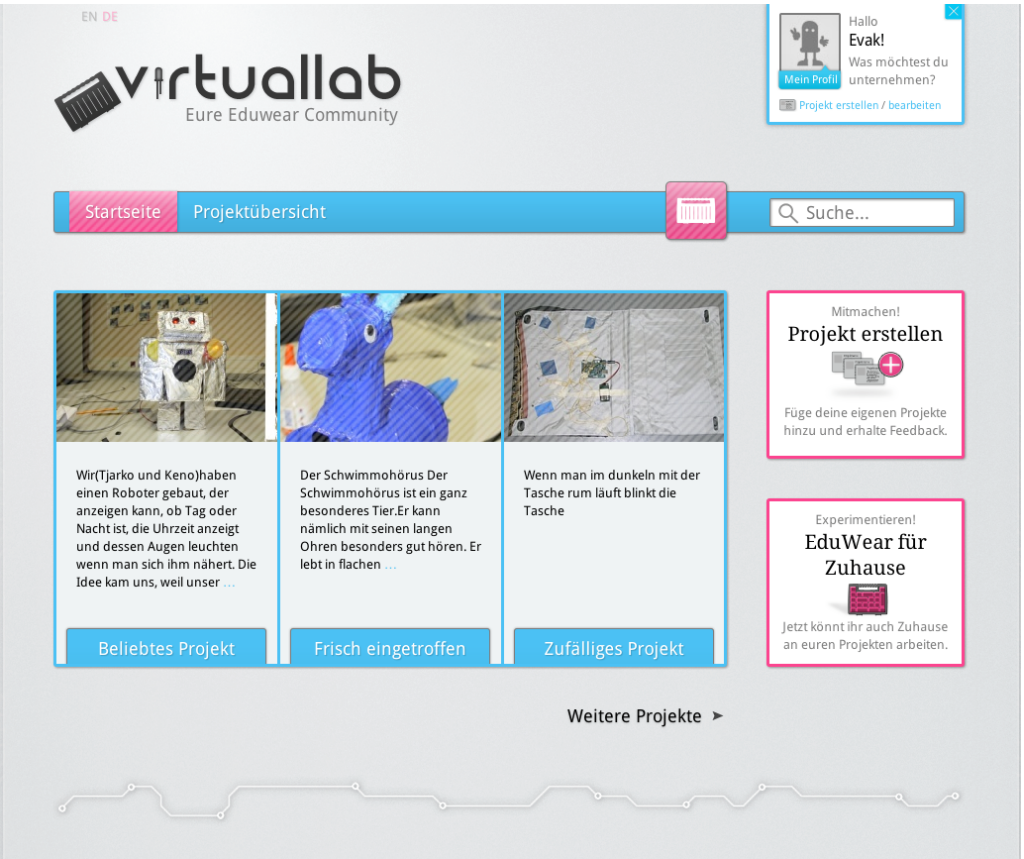


Abbildung 5.1: VirtualLab Startseite (März 2012)

5.1.3 Schlussfolgerungen zu Dokumentationstools für Maker_innen

Eine Projektdokumentation oder ein How-to im Nachhinein zu erstellen ist relativ aufwändig. Es erfordert eine editorielle Nachbearbeitung und erfolgt losgelöst vom Modellierungsprozess des Artefakts. Fotos, die eingebunden werden, wurden ggf. mit einer externen Kamera erstellt und müssen erst transferiert und ggf. nachbearbeitet werden. Die kürzeren Projektdokumentationen bedeuten weniger Aufwand als das Erstellen eines How-tos, stellen aber nur das Endergebnis dar und vernachlässigen dessen Entstehungsprozess, insbesondere die nicht über den (als Dateianhang bereitgestellten) Programmcode erschließbaren Eigenschaften des Artefakts. Der Ansatz von HowDo, Anleitungen mit einem mobilen Gerät zu erstellen, mit dem Fotos erstellt und unmittelbar hochgeladen werden können, scheint hier für das Abbilden während des Prozesses praktisch, erfordert im Falle von stofflich-digitalen Artefakten aber wiederum ein zusätzliches Transferieren des Programmcodes und anderer dazugehöriger Dateien.

Das Erstellen von How-tos oder von Projektdokumentationen mit den besprochenen Tools folgt einem einfachen, leicht verständlichen Schema einer chronologischen

Aneinanderreihung einzelner Projektschritte. Sie konzentrieren sich darauf, ein Projekt im Nachhinein abzubilden und eine für andere nachvollziehbare und zielführende Anleitung zu erstellen. Mit How-tos und ergebnisorientierten Projektdokumentationen werden Modelle der Modellierungsprozesse erstellt, die wiederum in Form der einzelnen Schritte oder Artefaktaspekte Teilmodelle enthalten, z.B. Programmcode oder Skizzen der Konstruktion. Im Gegensatz zu den Zielen dieser Arbeit – dem Abbilden von stofflich-digitalen Artefakten während des Modellierungsprozesses und damit Projekten, die noch nicht abgeschlossen sind – erscheint diese lineare Struktur nicht optimal geeignet. Sie widerspricht einem kreativen, lösungs-orientierten Schaffensprozess, in dem u.U. verschiedene alternative Lösungswege ausprobiert werden müssen, um zum gewünschten Ziel zu gelangen. Es lässt sich für diese Arbeit folgern, dass How-tos ein geeignetes Medium sind, mit dem Amateur_innen während der Modellierung stofflich-digitaler Artefakte entstehende Modelle abbilden können. Jedoch sollte auch nicht-lineares Vorgehen berücksichtigt werden und eine direktere Einbindung insbesondere von Fotos möglich sein.

Wie in Abschnitt 2.5.4 bereits angesprochen wurde, enthalten viele How-tos von Arduino-Projekten Programmcode und legen damit die abstrakten Modelle des Artefaktverhaltens offen. Es wird aber in der Regel kein unmittelbar nachvollziehbarer Bezug zwischen wahrgenommenem Verhalten und Code dargelegt. In Anlehnung an die Ausführungen in Abschnitt 2.5.4 kann gefolgert werden, dass How-tos und damit Dokumentationstools so gestaltet sein sollten, dass sie abstrakte Prozesse verständlich machen, aber auch so, dass die dort dokumentierten Artefakte tatsächlich realisiert werden können, so dass etwas über das Wesen Digitaler Medien durch eigene kompetente Umsetzung erfahren werden kann. Daher wäre aus Sicht dieser Arbeit eine engere Verknüpfung der digitalen Modelle mit ihrem wahrgenommenen, eher unformalem Erscheinen wünschenswert.

5.2 Allgemeinverständliches Modellieren von Interaktion

In Abschnitt 4.1 wurde beschrieben, dass auch unformale Modellierungstechniken beim Entwickeln informatischer Systeme zum Einsatz kommen können. Insbesondere, wenn Menschen ohne technische Ausbildung beteiligt werden sollen, sind solche Herangehensweisen von Vorteil. Aber auch um soziale Kontexte zu beschreiben oder Ideen und Visionen in einem designartigen Prozess zu entwickeln und zu erproben, können sie nützlich sein. Deshalb sind auch Tools aus Designkontexten von Interesse, die ein unformales designartiges Herangehen an das Entwickeln stofflich-digitaler Artefakte ermöglichen können, das eher die Ausgestaltung der Interaktion und User Experience fokussiert als deren Implementierung. Ein beliebtes Mittel, um Interaktion unformal abzubilden und zu visionieren, sind Szenarien und Storyboards. Das Erstel-

len von Storyboards hat außerdem strukturelle Ähnlichkeit mit den in Makerkontexten etablierten How-tos.

5.2.1 Interaktionsgestaltung mit Storyboards

Storyboarding ist ein aus dem Filmbereich entlehntes Modellierungsprinzip, das in der HCI aufgegriffen wurde. Truong, Hayes und Abowd (2006) definieren ein Storyboard als „a short graphical depiction of a narrative“ (Truong et al., 2006, S. 12). Ein Storyboard bildet durch eine Folge von chronologisch sortierten Bildern, die mit Untertexten versehen sind, ein Geschehnis ab. Solche narrativen Szenarien und daraus erstellte Storyboards sind nicht nur Kindern durch Bilderbücher oder Comics bekannt, sondern können generell als unformale Methode angesehen werden (Kühn, Keller & Schlegel, 2011).

Aus Sicht der AMT (siehe Abschnitt 3.1) handelt es sich bei Storyboards um reduzierte Abbildungen: Sie sind einerseits temporär reduziert, indem nur Schlüsselbilder gezeigt werden. Andererseits zeigen sie idealerweise reduzierte Abbildungen einer Szene und enthalten nur für diese relevante Merkmale. Ergänzt wird das einzelne Bildmodell durch abundante textuelle Modelle. Ein Storyboard ist ein zwei-dimensionales Modell eines zeitlichen Ablaufs, das diverse Modelle von Szenen enthält. Durch die Kameraperspektive auf eine Szene sehen die Betrachtenden diese aus Sicht der Urheber_innen des Storyboards. Deren Intentionen werden so allerdings nicht explizit vermittelt.

In der HCI wurden Storyboardmethoden ursprünglich eingesetzt für ‚walkthroughs‘ statischer Bildschirmzustände (z.B. bei Andriole, 1989). Seit Computersysteme nicht mehr nur die klassischen Interfaces aus Maus, Bildschirm und Tastatur aufweisen und der Nutzungskontext wichtiger geworden ist (siehe Abschnitt 4.1), werden auch Formen des Storyboarding präsentiert, die der klassischen Film-Variante mehr ähneln, indem sie nicht nur das Bildschirmbild beschreiben, sondern auch den Kontext der Nutzenden. Die wachsende Bedeutung mobiler Endgeräte und der User Experience rücken den Kontext der Nutzenden verstärkt in den Fokus. So reicht es nicht mehr aus, in einem Storyboard nur Interaktionswege als Bildschirmdraufsicht abzubilden.

Cilella, Berman und Rheinfrank (2010) beispielsweise bezeichnen mit Storyboarding eine Methode, mit der die User Experience von TUIs gestaltet wird. Storyboarding wird dazu in den Ideen- und Konzeptphasen eingesetzt. Dazu malt sich die Zielgruppe das Interagieren mit dem zu entwickelnden System anhand einer Erzählung aus, die mit professioneller Unterstützung in ein comicartiges Storyboard übertragen wird.

Tools für Storyboarding

Diverse Tools wurden entwickelt, die Interaktionsdesign mit Storyboarding unterstützen wollen.

Pic-up von Wahid, McCrickard, DeGol, Elias und Harrison (2011) ist ein Storyboarding-Tool, das Designer_innen beim Wiederaufnehmen bestehender Ideen und beim Kommunizieren dieser mit Hilfe von Storyboards als visuellen Erzählungen unterstützen will. Diese Storyboards unterscheiden sich von anderen Tools dadurch, dass sie nicht gezeichnet werden, sondern dass virtuelle Karten mit vorhandenen Bildern benutzt und zu Storyboards arrangiert werden (Wahid et al., 2011). Pic-up ist ein interessantes Beispiel, das das Sammeln und Wiederaufnehmen von Ideen im Designprozess unterstützt und mit Storyboards kombiniert.

Das Tool *Sketchify* von Obrenović und Martens (2011) orientiert sich an den Praktiken von Designer_innen wie sie in Abschnitt 4.3 beschrieben wurden. Dadurch ergänzt Sketchify die Tätigkeit des Skizzierens von User Interfaces durch exploratives Handeln mit interaktiven Materialien. Das Tool erlaubt es, Ein- und Ausgabegeräte und Schnittstellen zu anderen Entwicklungsumgebungen zu integrieren und bietet verschiedene Ebenen der Programmierung an, um die Geräte zu konfigurieren. Ziel ist es, Designenden in einer frühen Phase zu ermöglichen, die Interaktion und User Experience beim Entwickeln neuartiger Interfaces zu erleben und die Beschränkungen und Möglichkeiten der entsprechenden Technologien zu erfahren. Das System unterstützt die Anbindung vorhandener Tools und Umgebungen der Designenden. Sketchify richtet sich an professionelle Interaktionsdesigner_innen und orientiert sich an deren Praxis (Obrenović & Martens, 2011). Wenngleich das System auf den ersten Blick aufgrund der benutzten Techniken auch anwendbar für Amateur_innen zu sein scheint, fehlt diesen möglicherweise die Strategie und Methodik, das Tool effektiv zu nutzen. Außerdem konzentriert es sich mehr auf das Explorieren von Möglichkeiten der User Experience als das Umsetzen der Ideen, wozu ggf. auch weitere technische und informatische Fähigkeiten notwendig wären.

Greenberg, Carpendale, Marquardt und Buxton (2012) haben ein Arbeitsbuch herausgebracht, das Interaktionsdesigner_innen anleitet, narrative Storyboards zu erstellen, die Interaktionen von Nutzenden mit z.B. mobilen Geräten und deren Kontext beschreiben. Die Anleitungen erklären das Zeichnen von Storyboards von Hand und das Erstellen von Storyboards auf Basis von Fotos. Greenberg et al. (2012) betonen die Bedeutung des Nutzerkontexts und befürworten Storyboards, um diesen zu erfassen und zu antizipieren. Diese Technik stellt daher ein Beispiel dar, wie sozialer Kontext in Entwicklungsprozessen einbezogen werden kann, um Modellbildungskonflikte (siehe Abschnitt 3.3) zu berücksichtigen.

Das Prinzip des Storyboarding wird auch mit formalen Methoden verknüpft. Kühn et al. (2011) präsentieren einen methodischen Ansatz für die Entwicklung ubiquitärer Systeme, bei dem so genannte modellgestützte Storyboards erstellt werden. Durch

klassisches Storyboarding sollen erste Systemanforderungen erhoben werden, die Benutzende und deren Interaktionshandlungen erkennen lassen und insbesondere umgebungsbedingte Einflüsse auf kontextsensitive ubiquitärer Systeme berücksichtigen. Von diesen werden in einem iterativen Vorgehen Interaction- oder Use-Cases abgeleitet. Zielgruppe sind interdisziplinäre Entwicklerteams (Kühn et al., 2011). In der professionellen Softwareentwicklung finden sich auch Ansätze, die Storyboarding-Prinzipien für konzeptuelle Modellierungsprozesse mit formalen Modellen einsetzen (z.B. bei Singh, 2010; Beyer & Hassan, 2006; Bellamy et al., 2011). Diese Storyboards, die nach formalsprachlichen Regeln erstellt werden, ähneln meist Zustandsdiagrammen mit gerichteten Pfeilen, um Aktionen, Abhängigkeiten und Relationen abzubilden. Durch ihre formale Diagrammstruktur und symbolische, abstrakte Elemente sind sie als weniger allgemeinverständlich einzustufen.

Verwandte Ansätze finden sich bei Methoden für agile Softwareentwicklung. So wird beim Extreme Programming (XP) mit User Stories und Storycards (Beck, 2003) gearbeitet, mit denen Anforderungen aus ‚unformaler' Nutzersicht beschrieben werden. Sie dienen dazu, Aufgaben zu gliedern, um kurze Iterationszyklen zu ermöglichen. Eine Storycard beim XP sollte eine vom Kunden gewünschte Funktion des Systems enthalten (Beck, 2003). User Stories sind dynamischer als Storycards und bilden eher ein Arbeitspaket, das der Risikoanalyse, Aufwandschätzung und Iterationsplanung dient, als einen Teil der Spezifikation (Hanser, 2010). Sie werden im Rahmen von Planungsspielen erstellt, an denen neben Entwickelnden z.B. auch Vertreter_innen der Geschäftsleitung beteiligt sein sollen (Beck, 2003). User Stories oder Storycards dienen vornehmlich dazu, Anforderungen an das gesamte System festzuhalten, diese zu kommunizieren und zu planen. Die Umsetzung dieser – also das daraus entwickelte Interaktionsdesign und dessen Implementierung – bilden sie nicht direkt ab.

Szenarien und deren Darstellung

Storyboards sind durch ihre erzählende Struktur eng mit Szenarien verwandt und bieten eine Darstellungsform für diese. Szenarien werden in der HCI eingesetzt, um die Auswirkungen eines Systems auf die zukünftigen Nutzenden zu veranschaulichen und zu antizipieren (Rosson & Carroll, 2002).

Szenarien können von verschiedener Detailliertheit sein, abhängig davon, was für die aktuelle Aufgabe angemessen erscheint. Sie sollten dabei wesentliche Merkmale enthalten (Holbrook, 1990). Somit sind sie reduzierte Abbildungen, können also als Modelle im Sinne der AMT (siehe Abschnitt 3.1) angesehen werden. Für gewöhnlich wird ein Set aus mehreren Szenarien entwickelt. Daraus werden dann allgemeine Anforderungen für das System abgeleitet. Szenarien können als Text aufgeschrieben werden, sie können als Storyboards oder als Diagramme notiert werden oder sie können wie Rollenspiele live agiert werden (Carroll, 1997).

Szenarien zu agieren, also darstellerisch zu spielen, ist eine Methode, die im Interaktionsdesignkontext angewandt wird. Das Vormachen dient dabei zum einen zur Visionierung, aber auch zur Kommunikation und zum Testen von Ideen (Iacucci, Iacucci & Kuutti, 2002; Macaulay et al., 2006). Solche Methoden werden in verschiedenen Bereichen eingesetzt, werden aber insbesondere dann interessant, wenn z.B. Interaktionen mit mobilen Geräten gestaltet werden, die stärker von kontextuellen Faktoren geprägt sind und einen höheren körperlichen Einsatz erfordern als z.B. Software Interfaces an stationären Computern (z.B. Kuutti, Iacucci & Iacucci, 2002; Oulasvirta, Kurvinen & Kankainen, 2003). Damit sind Darbietungen (engl. performances) allerdings temporäre dynamisch-flüchtige Modelle. Sie für Designzwecke zu nutzen ermöglicht es, Szenarien körperlich auszuagieren und dabei vermehrt implizites Wissen wahrnehmbar zu machen. Performances sind eine unformale, natürliche Methode, kosten die Vorführenden aber ggf. Überwindung. Um sie als Modelle im Sinne externer Repräsentationen zu nutzen, müssen sie festgehalten werden, z.B. durch eine Videoaufnahme oder – reduzierter – als Storyboard.

Auch bei der Technik der Stop-Motion-Animation werden mittels Szenarien Interaktionen erprobt und in Bildern festgehalten. Diese Technik wird sonst bei Trickfilmen angewandt, Bonanni und Ishii (2009) und Fallman und Moussette (2011) schlagen sie für das Interaktionsdesign vor. Die Abfolge von dabei aufgenommenen Bildern, die beim Interaktionsdesign nicht als Film ablaufen müssen, sondern eher aus Schlüsselbildern bestehen, erinnern an Storyboards. Eingesetzt werden soll diese Technik zur Entwicklung von Low-Fi-Prototypen in der Ideenentwicklung, um Interaktionsszenarien zu entwerfen. Fallman und Moussette (2011) sehen die Technik als besonders geeignet an, um fließende Abläufe auf einem Interface und lückenlose Übergänge im Umgang mit einer Ubiquitous Computing-Umgebung detailliert abzubilden und zu durchdenken. Als Gegenstände dienen Alltagsmaterialien (Fallman & Moussette, 2011) oder auch Modelliermasse und biegbare Figuren (Bonanni & Ishii, 2009). Die Methode eignet sich laut Bonanni und Ishii (2009) für Anfänger_innen wie für Expert_innen, jedoch weniger zur Darstellung von Sachverhalten, die nicht visuell selbsterklärend sind.

Schlussfolgerungen zu Storyboards und Szenarien

Szenarien und Storyboarding haben sich in den vergangenen Jahren im nutzerzentrierten (user-centered) Interaktionsdesign insbesondere bei der Entwicklung ubiquitärer Systeme etabliert (u.a. Bonanni & Ishii, 2009; Kühn et al., 2011; Greenberg et al., 2012; Rosson & Carroll, 2002). Gründe dafür sind, dass sich mit Storyboards der bei ubiquitären Systemen wichtige Kontext abbilden lässt und dass sie durch ihre Informalität von allen am Prozess Beteiligten erstellt und verstanden werden können.

Storyboards können helfen, kontextuelle Faktoren einzubeziehen und so den in Abschnitt 3.3 beschriebenen Konflikten entgegenzuwirken. Gleichzeitig sind sie für das

User Experience Design mobiler oder tangibler Systeme geeignet. Die große Bandbreite verschiedener Anwendungsgebiete und Zielgruppen im Interaktionsdesign, in der diese Methode eingesetzt wird, zeigt, dass dies ein in struktureller Hinsicht allgemeinverständlicher und flexibler Ansatz ist, der sich für verschiedene komplexe Inhalte eignet. Auch die populäre Dokumentationsweise von Makerprojekten, die so genannten How-tos, sind strukturell wie Storyboards aufgebaut. Sie werden – im Vergleich zu Storyboard-Techniken in der HCI – nicht zum Planen, sondern erst im Nachhinein oder während der Konstruktion eines Artefakts erstellt.

Nach- und Vorteil von Storyboards für diese Arbeit liegen nah beieinander: Sie bieten lose Rahmenbedingungen, wodurch sie sich für verschiedene Vorgehensweisen eignen. Zugleich mag diese lose Rahmung für manche Tätigkeiten zu wenig Anhaltspunkte bieten, um wesentliche Eigenschaften von Modellen mit dieser Technik abzubilden.

Storyboards konzentrieren sich meist auf die Interaktion von Menschen bei der Nutzung informatischer Systeme. Da mit Storyboards kontextuelle Faktoren, d.h. pragmatische Merkmale, gut abgebildet werden können, erscheint dieses Prinzip nachahmenswert. Wie schon bei den How-tos beschrieben, steht die klassische lineare Struktur einem Abbilden von Interaktion als auch einem aktuellen Prozess des Selbermachens entgegen. Da Storyboards sich meist auf Interaktionsaspekte sowie damit verbundene pragmatische Merkmale konzentrieren, ist zudem zu überlegen, wie diese mit abstrakten Modellen zu verbinden sind, die notwendig sind, um ein stofflich-digitales Artefakt umfassend abzubilden.

5.2.2 Mitteilung pragmatischer Modellmerkmale

In Abschnitt 4.4.4 wurde thematisiert, dass auch im Design Konflikte entstehen, weil Designer_innen und Nutzende nur durch das Artefakt kommunizieren, in dem Nutzende die von den Designer_innen konzipierte Verwendung erkennen müssen. Aber auch, wenn an einem Designprozess mehrere Parteien beteiligt sind, kann es sinnvoll sein, während des Designprozesses Intentionen und Entscheidungen – pragmatische Modellmerkmale – zu dokumentieren und gegenseitig zu kommunizieren. Die meisten der bisher vorgestellten Tools legen nicht explizit Wert darauf, pragmatische Merkmale von Modellen abzubilden, wie z.B. Intentionen.

Ein Ansatz aus dem Designbereich, der sich dem Dokumentieren solcher Merkmale widmet, sind „annotated portfolios“ (Gaver & Bowers, 2012, S. 40). Mit dieser Methode sollen Designende ihre Intentionen und Entscheidungsgründe bei der Entwicklung von Artefakten dokumentieren und mitteilen können. Das Konzept beruht ebenfalls auf dem Text-mit-Bild-Prinzip, wie es für die Dokumentation von Makerprojekten oder auch in Storyboards verwendet wird. Bei den annotated portfolios werden Bilder von Produkten oder Prototypen durch Textannotationen ergänzt, in denen die Entwerfenden Auskunft über Hintergründe ihrer Entscheidungen geben. Gaver und

Bowers (2012) betrachten dieses Konzept als Beitrag zur Designforschung, um Entscheidungsprozesse und Vorgehensweisen nachvollziehbar darzustellen.

Wie in Abschnitt 3.3.5 bereits erwähnt, wurden für Ingenieurtätigkeiten so genannte Design-Rationale-Systeme entwickelt, mit denen getroffene Entscheidungen und ihre Beweggründe, die zu einem Artefakt geführt haben, dokumentiert werden sollen. Vor allem in den 1990er Jahren wurden Design-Rationale-Systeme in der HCI-Forschung thematisiert. Design rationale bezeichnet im Kontext von Ingenieurdisziplinen die Darlegungen der während des Entwicklungsprozesses[5] getroffenen Entscheidungen, Beweggründe und Abwägungen, die in üblichen Systemspezifikationen weniger enthalten sind. Damit können Dokumentationen, die als design rationale erstellt werden, z.B. bei der späteren Wartung und Umgestaltung der Artefakte nützlich sein (Regli, Hu, Atwood & Sun, 2000). In einem umfassenden Überblick über das Gebiet definieren Regli et al. (2000, S. 210): „(...), a design rationale system intends to let designers think and discuss design within a certain knowledge representation framework."

Ein Design-Rationale-System zeichnet aktuelle Erkenntnisse bzw. Ereignisse auf und erstellt daraus die design rationales als einen strukturierten Auszug der Rohdaten. Die Strukturen, in denen die Systeme design rationales erfassen, werden als formal oder semi-formal beschrieben. Die Systeme werden oft mit CAD-Systemen oder CASE-Tools kombiniert. Entscheidungen und Beweggründe werden entweder von den Entwickelnden selbst dokumentiert und/oder durch das System automatisch erfasst (Regli et al., 2000). Regli et al. (2000) unterscheiden zwischen Feature-orientierten und Prozess-orientierten Systemen. Feature-orientierte Systeme kommen bevorzugt bei standardisierten Entwicklungsprozessen zum Einsatz. Sie konzentrieren sich auf die Darstellung des Artefakts und basieren auf einem Logik-basierten Regelsystem, das den Entwicklungsprozess leitet. Sie sind damit formaler als die Prozess-orientierten Systeme, die sich für offenere und weniger regelgeleitete Entwicklungsprozesse anbieten und vorwiegend deren Abläufe abbilden (Regli et al., 2000). Trotzdem bilden Prozess-orientierten Systeme Entwicklungsprozesse formalisiert z.B. als Graphen mit Knoten ab, die dann (i.d.R. vom System) strukturiert und verlinkt werden entsprechend der getroffenen Entscheidungen. Bei den verknüpfbaren Elementen handelt es sich z.B. um Fragen des Systems und dazugehörige Antworten der Nutzenden oder auch Multimediadateien oder Email-Konversationen (z.B. bei Shipman & McCall, 1997). Regli et al. (2000) weisen darauf hin, dass es nur wenige Berichte über erfolgreiche Design-Rationale-Systeme gibt.

Wenngleich Design-Rationale-Systeme pragmatische Merkmale adressieren, so sind sie eher für professionelle Ingenieurbereiche gemacht, in denen nach etablierten und eher strukturierten und standardisierten Methoden vorgegangen wird. Die darin abgebildeten Entwicklungswege sind mitunter nur für Fachleute verständlich und eher auf

[5] Um den englischen Begriff ‚Design' von der in Kapitel 4 intendierten Bedeutung abzugrenzen, wird er hier mit *Entwicklung* übersetzt.

Teamarbeit ausgelegt. Insbesondere, wenn wir es mit einem unstrukturierten iterativen Vorgehen zu tun haben, kann dies schwierig werden bzw. gegen die Gewohnheiten der Nutzenden sprechen.

5.3 Informatisches Modellieren stofflich-digitaler Artefakte mit konkretem Zugang

In Abschnitt 4.5 wurde das Konzept der Übersetzungen und der Erfahrungsorientierung herangezogen. Dieses zeigt, dass es für ein Verständnis komplexer Systeme für Laien hilfreich sein kann, wenn abstrakte und konkrete Repräsentationen derselben Systemsichten miteinander in Beziehung gebracht und wechselseitig verändert werden können.

An dieser Stelle werden nun Tools für Menschen, die keine oder wenig Erfahrung im formalen Modellieren haben, vorgestellt, mit denen stofflich-digitale Artefakte modelliert bzw. programmiert werden können. Sie ermöglichen ein unformales designassoziiertes Herangehen an den Umgang mit stofflich-digitalen Artefakten und machen sich konkrete Zugänge zu Nutze, um komplexe technische stofflich-digitale Systeme und deren formale Modelle in Bildungskontexten zugänglicher zu machen. Vorherrschend sind hierbei Konzepte, die Ebenen verschiedener Stufen der Abstraktion verknüpfen. Sie stammen aus den Bereichen technischer beruflicher Ausbildung, Prototyping im Design und Making.

5.3.1 Von Performance zu Programmierung: MoTo

Mit *MoTo* (Katterfeldt & Schelhowe, 2008; Katterfeldt, 2007) wurde von der Autorin ein Modellierungstool konzipiert und entwickelt, das auf dem Storyboarding-Prinzip beruht. Es soll Kindern und Jugendlichen beim Programmieren von Cricket-Mikrocontrollern[6] helfen, den Übergang vom angedachten Verhalten zur Programmierung zu bewerkstelligen.

Die wesentlichen Konzepte von MoTo sind die informellen Methoden „reverse storyboarding“ (Katterfeldt & Schelhowe, 2008, S. 225) und „acting out“. MoTo bezieht den Hardwarekontext und damit das konstruierte Artefakt ein und begleitet die Nutzenden damit schrittweise durch den Modellierungsprozess.

Dem Konzept von MoTo liegen Überlegungen aus Konstruktionismus, Informatikdidaktik sowie Modelltheorien zugrunde (vgl. dazu Katterfeldt, 2007). MoTo baut auf den Prinzipien von Szenarien oder ‚Simulacra‘[7] und Storyboards auf. Dabei handelt es sich um Tätigkeiten und Designtechniken, mit denen schon Kinder vertraut sind. Die Nutzenden erstellen auf der ersten Ebene von MoTo ein Szenario ihrer Projek-

[6] `http://www.handyboard.com/cricket/` (aufgerufen am 24.02.2013)

[7] Von einem menschlichen Akteur ausgeführte Simulation, vgl. Ackermann (2004).

tidee, machen diese vor und filmen sich dabei. Anschließend erstellen sie aus dem Video ein Storyboard. Die Storyboardelemente werden auf der nächsten Ebene in eine Diagramm-ähnliche, sich verzweigende Struktur umgeordnet. Auf der dritten Ebene wird schließlich ein visuelles, Diagramm-ähnliches Programm erstellt (aus dem textueller Logo-Programmcode generiert werden kann). Diese Aktivitäten bauen aufeinander auf und erhöhen jeweils den Abstraktionsgrad. Wenn auf eine darüberliegende Ebene gewechselt wird, kann die vorherige Ebene beliebig ein- und ausgeblendet werden. Dadurch können Nutzende die vorherige, weniger abstrakte Ebene als Vorlage benutzen.

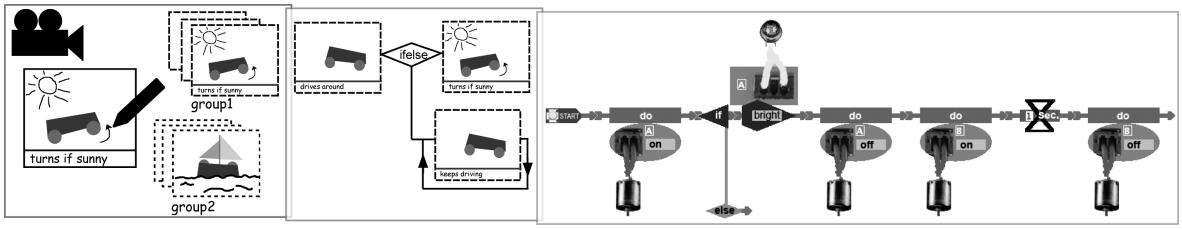


Abbildung 5.2: Phasen der Modellierung mit MoTo (aus Katterfeldt & Schelhowe, 2008)

Zuerst das Verhalten des Artefakts zu filmen und aus diesem Video dann Schlüsselbilder zu generieren, ist dem Ablauf beim klassischen Storyboarding entgegengesetzt. Durch den Vorgang des Vormachen des Verhaltens des Artefakts werden diese Handlungen ‚be-greifbar' als enaktive Repräsentationen (Abbildung 5.2 links).

Auf der zweiten Ebene werden Storyboardkarten mit Kontrollstrukturen kombiniert. Diese Elemente brechen die traditionell lineare Storyboard-Struktur auf und geben damit den Nutzenden einen ersten Eindruck, dass Programme anders als Erzählstrukturen oder Storyboards nicht linear sein müssen (Abbildung 5.2 Mitte).

Auf der Programmierebene werden die konkreten Bilder und Beschreibungen der vorherigen Ebenen durch Programmierbausteine ersetzt. Um eine Brücke zwischen virtueller Programmierung und konkreten Hardwareelementen zu schaffen, wurden letztere durch Symbole in die visuellen Programmierbausteine integriert (Abbildung 5.2 rechts).

In Anwendungstests waren alle Kinder in der Lage, mit MoTo ihre Projekte umzusetzen. Die Äußerungen und das Vorgehen der Nutzenden deuteten darauf hin, dass durch das acting out die umzusetzenden Interaktionskonzepte in einer frühen Projektphase mehr durchdacht wurden als in den Workshops zuvor. Diejenigen Kinder, die bereits Erfahrung in Workshops gesammelt hatten, wiesen aber darauf hin, dass sie das Tool nicht mehr bräuchten, sondern es eher für Anfänger hilfreich sei (Katterfeldt, 2007). MoTo ist also nur bedingt geeignet für fortgeschrittene Projekte und Nutzende.

MoTo ist ein Storyboard-basiertes Tool, das unformales Storyboarding, Vormachen und Programmieren (also formalisierendes Vorgehen) in einem Prozess und Tool ver-

eint. Damit ist es auch eine Visuelle Programmierumgebung und enthält eine Visuelle Programmiersprache, mit der formale Modelle erstellt werden können.

5.3.2 Gegenständlicher Zugang zur Modellierung komplexer Systeme

In Abschnitt 4.5 wurde bereits auf die Beziehungen zwischen stofflich-digitalen Artefakten, Tangible User Interfaces und Mechatronik-Systemen hingewiesen und relevante Konzepte aus letzterem Gebiet vorgestellt. An dieser Stelle sollen Systeme betrachtet werden, die diese Konzepte implementieren und für Lernkontexte entwickelt wurden, um Anfänger_innen über konkrete, stoffliche Interfaces einen erfahrungsorientierten Zugang zu formalen Modellen anzubieten.

Hornecker et al. (2001) präsentieren das System RUGAMS für rechnergestützte Übergänge zwischen gegenständlichen und abstrakten Modellen produktionstechnischer Systeme zur Planung von Förderbandanlagen. Es umfasst die Modellierung des statischen Aufbaus als auch des dynamischen Verhaltens der Anlage. Die Objekte (hölzerne Bausteine als abstrakte Repräsentationen oder Fischertechnik-Komponenten mit eher ikonischem Charakter) werden im virtuellen Modell so platziert, wie sie im gegenständlichen Modell bewegt wurden. Das dynamische Verhalten kann programmiert werden, indem es durch das Bewegen von Objekten demonstriert wird. Durch Token können Verzweigungsregeln gekennzeichnet werden. Durch das Vormachen wird ein Petri-Netz-artiges abstraktes Modell erzeugt, das wiederum automatisch in Code übersetzt wird. Dieser generiert eine virtuelle Simulation und steuert die gegenständlichen Fischertechnik-Modelle, die Aktuatoren und Sensoren enthalten. In die virtuelle Simulation werden auch Bilder der gegenständlichen Anlage einbezogen (Schäfer & Bruns, 2001; Hornecker et al., 2001).

Ein ähnliches Tool ist eine Pneumatik-Lernumgebung für Berufsschulen. Um Schüler_innen zu erleichtern, verschiedene Repräsentationsebenen in Bezug zu bringen, wird auch hier eine gegenständliche Lernumgebung mit computergenerierten Modellen verknüpft. In Echtzeit während des Schaltungsaufbaus wird das virtuelle Modell erstellt, so dass dieses direkt auf das aktuelle gegenständliche bezogen werden kann (Bruns, 2000; Hornecker et al., 2001).

Beide Tools implementieren Konzepte der Erfahrungsorientierung und der Übergänge, d.h. der automatischen Übersetzungen mit enger Interaktionskopplung zwischen realem und virtuellem Modell (siehe Abschnitt 4.5.2). Hornecker et al. (2001) weisen darauf hin, dass bei solchen Systemen immer eine Formalisierung notwendig ist, was technische Beschränkungen mit sich bringt bei den gegenständlichen Interaktionsmöglichkeiten.

Die vorgestellten Systeme verbinden konkrete Modelle mit abstrakten Modellen und schaffen so neue Zugänge insbesondere für Anfänger_innen. Jedoch handelt es sich um spezielle, sehr komplexe Systeme, die eine aufwändige Infrastruktur erfordern. Im Makerkontext können wir solche Ausstattung nicht voraussetzen. Die Übersetzung

zwischen konkreten und abstrakten Modellen bzw. Repräsentationen müssen die Anwendenden hier vorläufig selbst leisten. Dennoch ist das Verbinden von ikonischen und symbolischen Elementen, die an Erfahrungen anknüpfen und das Verständnis erleichtern, ein interessanter Ansatz für das Gestalten von Tools für Amateur_innen. Dabei könnte es hilfreich sein, wenn abstrakte Informationen durch ikonische Abbildungen unterstützt werden, die technische Komponenten gegenständlich und so zeigen, wie sie von den Benutzenden wahrgenommen werden.

5.3.3 Prototyping-Systeme im Designkontext

An der Entwicklung von HCI-Systemen wie stofflich-digitalen Artefakten sind heutzutage auch Menschen ohne technische Ausbildung beteiligt. Wie in Abschnitt 4.4 beschrieben, nutzen Designer_innen Modelle eher in Form von Mock-ups und Prototypen als Reflexionsobjekt für die Designidee. Daher ist es beim Design interaktiver Artefakte für Designer_innen wichtig, schnell und einfach auch interaktive Prototypen erstellen zu können, um Interaktionsmöglichkeiten zu explorieren. Trotz allem erfordert das Erstellen interaktiver Prototypen das Erstellen eines oder mehrerer formaler Modelle, indem die Hardware konfiguriert und die Interaktionsabläufe programmiert werden müssen. Tools, die dies leisten wollen und sich an Designer_innen oder Amateur_innen richten, knüpfen oft an eher unformal erscheinende Praktiken an oder versuchen durch Automatisierung und black boxing (dem Verstecken komplexer Funktionen in Modulen) solche Modellierungsprozesse zu vereinfachen.

D.tools (Hartmann et al., 2006) ist ein Toolkit für das schnelle Erstellen von Hard- und Software-Prototypen. Mittels einer visuellen, auf State-Charts basierenden Oberfläche können auch technisch unerfahrene Designer_innen Prototypen erstellen. Gleichzeitig kann aber auch Programmcode editiert werden. Das System unterstützt das Erstellen Hardware-basierter Prototypen auf verschiedenen Ebenen, u.a. durch eine Schnittstelle zum Desktop-PC wie auch das Erstellen von Schaltkreisen. Mit D.tools können Prototypen nicht nur entworfen, sondern auch getestet und analysiert werden (Hartmann et al., 2006). D.tools ist somit für Rapid Prototyping gut geeignet und bietet verschiedene Schwierigkeitsstufen an, eignet sich aber weniger, um vollständige Projekte umzusetzen. Eine Erweiterung von D.tools stellt *Exemplar* (Hartmann, Abdulla, Mittal & Klemmer, 2007) dar. Damit werden Input-Eingaben z.B. durch Hardware-Sensoren in Echtzeit visualisiert, so dass Designer_innen diese durch Vormachen und anschließendes Editieren und Analysieren spezifizieren können. Damit sollen sich die Nutzenden auf den Designprozess konzentrieren können und müssen sich weniger mit technischen Details befassen. Ähnliche Tools, vorwiegend für Designer_innen zum Prototyping von Ubiquitous-Computing-Anwendungen mit graphischen Oberflächen, sind *Calder Toolkit* (Lee et al., 2004), *Voodoo Flash* (Spiessl, Villar, Gellersen & Schmidt, 2007) oder *Phidgets* (Greenberg & Fitchett, 2001).

Während solche Tools beim schnellen Umsetzen lauffähiger Prototypen sinnvoll sein können, so legen die meisten weniger Wert darauf, die abstrakten Modelle hinter der konkreten Interaktion und deren Zusammenhänge offen zu legen und verständlich zu machen. Durch die Ebenen mit verschieden abstrakten Modellen könnten die Nutzenden aber Einblicke in diese bekommen.

5.3.4 Modellierung von Hardwareaufbauten

Für Amateur_innen, die mit Arduino-Technologie Projekte erstellen, stehen diverse Softwaretools zur Verfügung. Diese unterstützen nicht nur die Programmierung, sondern auch den Aufbau von Schaltkreisen und Hardwarekonfigurationen.

Fritzing (Knörig, Wettach & Cohen, 2009) ist ein Tool, mit dem die Hardwarearchitektur von Arduino-Prototypen dokumentiert, eigene PCB-Layouts erstellt und mit anderen ausgetauscht werden können. Auch dieses Tool verfolgt ein Mehrebenenprinzip verschieden abstrakter Darstellungen. Mittels drag-and-drop kann ikonisch dargestellte Arduino-Hardware in der Software zusammen gesteckt werden (siehe Abbildung 5.3). Gleichzeitig wird dadurch die entsprechende schematisch-technische Grafik erstellt. Außerdem kann das Layout in Dateiformaten exportiert werden, die von PCB-Prägemaschine verwendet werden. Mit Fritzing können also Arduino-Projekte hinsichtlich ihres Hardware-Aufbaus anschaulich dokumentiert werden. Durch die verschiedenen Sichtweisen eignet sich das Tool für Anfänger_innen, während fortgeschrittene Nutzer_innen mit den symbolischen Modellen arbeiten können.

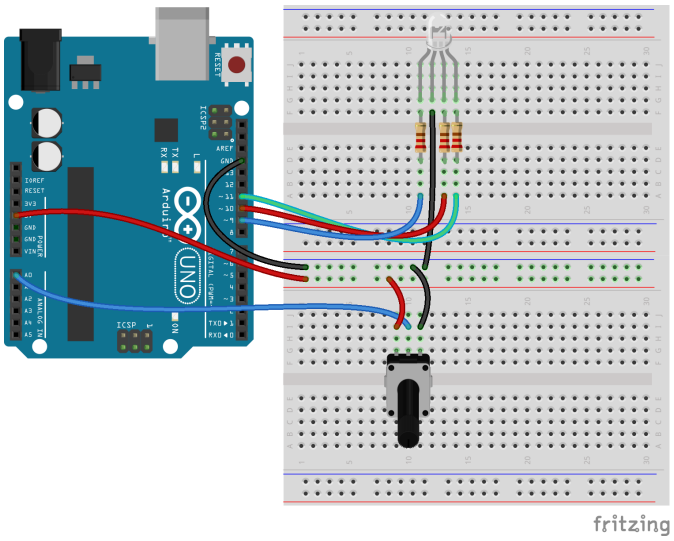


Abbildung 5.3: Schaltkreis mit Fritzing

Die Firma AutoDesk arbeitet mit *123D Circuits*[8] an einem vergleichbaren, kommerziellen Tool.

[8] `http://www.123dapp.com/sandbox` (aufgerufen am 05.10.2013)

5.3.5 Visuelle Programmiersprachen (VPLs)

Die meisten so genannten Visual Programming Languages (VPLs) benutzen eine nicht-textuelle Notation. Damit soll der Aufwand minimiert werden, die Syntax einer neuen Sprache und deren grammatikalische Regeln lernen zu müssen, um ein Programm erstellen zu können. VPLs nutzen zweidimensionale Symbole, die für textuelle Programmierkommandos stehen. Diese Blöcke können zusammengefügt werden zu einem ausführbaren Programm. Oft sind die Blöcke so gestaltet, dass sie nur eine mögliche Kombination von Blocktypen zulassen, um zu einem syntaktisch richtigen Programm zu führen (Kelleher & Pausch, 2005).

In den 1990er Jahren wurden VPLs für professionelle Programmierer_innen als zukunftsweisend angesehen. Man erhoffte sich davon, den kognitiven Aufwand beim Programmieren zu reduzieren (z.B. Whitley & Blackwell, 1997). Durchgesetzt haben sie sich jedoch nicht. Die in den vergangenen Jahren zur Programmierung von Mikrocontrollern erhältlichen VPLs wurden ursprünglich meist für Kinder entwickelt, die noch unsicher im Schreiben sind, so dass die Benutzung einer textuellen Sprache zusätzliche Hürden aufbauen würde. Aber auch für ältere Nutzende aus nicht-technischen Bereichen sind im Zuge der Makerbewegung VPLs in den vergangenen Jahren interessant geworden. So gibt es mittlerweile auch für die Programmierung von Mikrocontrollern wie Arduino diverse Tools, die sich an Amateur_innen ohne Programmiererfahrung richten. Diese benutzen in der Regel eine graphische Programmieroberfläche, in der Programmierkommandos in Form graphischer Blöcke aneinander gereiht werden. Diese werden vom Tool in textuellen Programmcode übersetzt.

Beim Interaktionsdesign ist das Erstellen eines vom Computer bzw. Mikrocontroller ausführbaren Programms bzw. Modells ein entscheidender Schritt im Modellierungsprozess. Spätestens mit Erstellen des Programmcodes als einem maschinell ausführbaren Modell ist ein großer Formalisierungsschritt notwendig, da Ausschnitte nicht-formaler Realität (bzw. Modelle dieser) in ein formales Modell überführt werden müssen.

Amici

Amici[9] bietet eine graphische Programmieroberfläche, mit der Mikrocontroller der Arduino-Familie durch das Aneinanderreihen graphischer Blöcke, die Programmierkommandos repräsentieren, programmiert werden können (Abbildung 5.4). Diese Block-Repräsentationen werden von der Software in textuellen Arduino-Code übersetzt. Die Programmierumgebung wurde in der Arbeitsgruppe dimeb – unter Mitwirkung der Autorin dieser Arbeit – entwickelt (mehr dazu in Abschnitt 6.4). Amici hat sich als einfacher Einstieg in die Programmierung von Arduino-Mikrocontrollern in vielen TechKreativ-Workshops bewährt. Rückmeldungen von Nutzenden lassen dar-

[9] `http://dimeb.de/eduwear` (aufgerufen am 02.03.2014)

auf schließen, dass Amici auch gern von Menschen benutzt wird, die eher künstlerisch interessiert sind und mit Amici einen leichten Einstieg finden.

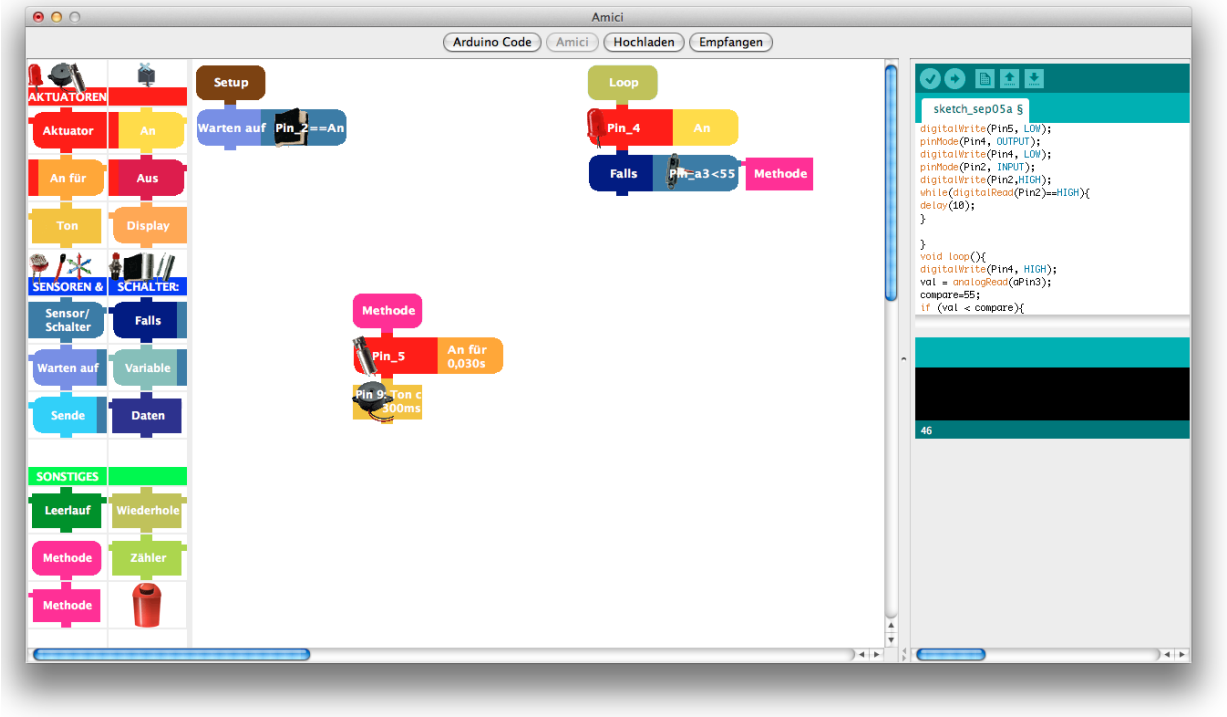


Abbildung 5.4: VPL Amici (Version amici0022p im März 2012)

Wenngleich sich Amici als IDE für einen Einstieg in das Programmieren bewährt hat, so bietet die graphische Programmierung Einschränkungen, wenn komplexere Programme erstellt werden sollen. Beispielsweise können Variablen nur in begrenztem Umfang benutzt werden; auch das Einbinden von Arduino-Bibliotheken lässt sich nicht flexibel vornehmen. Innerhalb von Amici kann auch in eine Code-Ansicht, die der normalen textuellen Arduino-IDE entspricht, gewechselt werden und mit dem textuellen Arduino-Programmcode weiterprogrammiert werden.

Die seit 2010 veröffentlichten Versionen von Amici enthalten Weiterentwicklungen, die in Zusammenhang mit dieser Arbeit getätigt wurden. Diese werden in Abschnitt 6.4.2 beschrieben und begründet. Unabhängig von der vorliegenden Arbeit wird Amici in TechKreativ-Workshops und von Interessierten weltweit zum Programmieren von Arduino-Boards genutzt.

ModKit (Millner & Baafi, 2011) ist eine Visuelle Programmierumgebung, die ähnlich wie Amici aufgebaut ist. Sie bietet aber neben der Block- und der Code-Ansicht noch eine Setup-Ebene, auf der die Hardware-Anschlüsse auf einer Abbildung eines Arduino-Boards konfiguriert werden können. Die graphische Blocksprache von Modkit basiert auf Scratch (s.u.).

Minibloq[10] ist eine weitere Visuelle Programmiersprache für Arduino und ähnelt in den wesentlichen Merkmalen Modkit und Amici.

Bei *S4A*[11] handelt es sich um eine Variante der Visuellen Programmierumgebung Scratch, bei der auch Ein- und Ausgaben von Arduino-Boards in Projekte einbezogen werden. Somit können die in Scratch üblichen Animationen um Interaktionen mit einem angeschlossenen Arduino-Board erweitert werden. S4A benötigt dazu die Installation einer speziellen Firmware auf dem Arduino-Board. In S4A ist ein Arduino-Board abgebildet, an dem dann in Echtzeit alle Sensorwerte beobachtet werden können.

Das oben vorgestellte Tool MoTo kann ebenfalls als VPL eingestuft werden, beschränkt sich aber auf die Programmierung von Cricket-Mikrocontrollern (siehe oben).

5.3.6 Einsichten über Tools zur formalisierenden Modellbildung

In diesem Abschnitt wurden Tools vorgestellt, die Menschen, die noch wenig Erfahrung auf dem Gebiet haben, beim Formalisieren von Modellen bzw. Programmieren unterstützen wollen. Die vorgestellten Tools für Designer_innen zielen vorwiegend darauf ab, möglichst einfach funktionale Prototypen zu erstellen, die grundlegende Funktionalität und Interaktionsmöglichkeiten erfahren lassen. Sie ermöglichen unformales (oder unformal erscheinendes) Modellieren. Die formalisierende Interpretation und Übersetzung der Eingaben erfolgt durch das System. Allerdings scheinen diese (Modellierungs-)Möglichkeiten die erforderlichen Formalisierungen für die Anwendenden eher zu verstecken. Deshalb wurden auch für Laien geeignete Programmiertools für Arduino-Boards betrachtet. Als Abbildungen sind visuelle Programme, die damit erstellt werden, relativ leicht verständlich und übersichtlich, d.h. mit wenigen Blicken erfassbar. Bezüge zu Hardwareelementen, also dem Artefakt, werden in manchen der Tools hergestellt. Durch das Wechseln zwischen verschieden abstrakten Repräsentationen, wie z.B. textuellem Code und ikonischen Blöcken, werden Übersetzungen der Modelle nachvollziehbar dargelegt. Es kann daher angenommen werden, dass Programme, die mit diesen Tools erstellt werden, für Laien eine Vorlage bilden, die sie eher verstehen und ändern können, als textueller Programmcode, der möglicherweise nur kopiert aber nicht verstanden wird.

5.4 Gestaltungsrichtlinien für Makertools

Bisher wurden in diesem Kapitel digitale Werkzeuge betrachtet, die bei verschiedenen Aspekten des Modellierens stofflich-digitaler Artefakte helfen können. Nun soll der Frage nachgegangen werden, welche übergeordneten Hinweise aus wissenschaftlicher Sicht bei der Konzeption von Tools zu beachten sind, die Maker_innen beim Selbermachen stofflich-digitale Artefakte unterstützen wollen.

[10] `http://blog.minibloq.org` (aufgerufen am 24.02.2013)
[11] `http://s4a.cat` (aufgerufen am 01.10.2013)

Resnick et al. (2005) präsentieren zehn „Design Principles for Tools to Support Creative Thinking“ (Resnick et al., 2005, S. 3). Sie fokussieren dabei auf Computersysteme, die zur Komposition, also dem Erstellen, Verändern, Interagieren und Spielen mit digitalen oder stofflichen Dingen genutzt werden können. Ihr Anliegen ist es, Anwender_innen zu befähigen und zu ermutigen, innovativ zu sein. Interessant ist, dass einige der Guidelines auch unter dem Titel „Some reflections on designing construction kits for kids“ von Resnick und Silverman (2005) veröffentlicht wurden – ein weiterer Hinweis auf die Nähe des konstruktionistischen Lernverständnisses zu den Tätigkeiten von Maker_innen und Designenden.

Nach Resnick et al. (2005) sollten Tools für kreatives Denken bzw. für Construction Kits Freiräume und Flexibilität bieten. Sie sollen beim Explorieren des Möglichkeitsraums helfen, aber auch dabei, Projekte verschiedener Art und Schwierigkeitsstufen zu bewältigen. Dabei sollen die Nutzenden ihren eigenen Vorlieben hinsichtlich des Vorgehens folgen können und ihre gewohnten Praktiken anwenden können. Auch sind die Zusammenarbeit und der Austausch von Projekten mit flexiblen Schnittstellen zu unterstützten. Gleichzeitig sollen solche Tools simpel sein. So genannte „black boxes“ (Resnick et al., 2005, S. 9) gilt es sorgsam zu wählen. Diese legen fest, inwieweit formalisierte, abstrakte Prozesse versteckt oder vereinfacht dargestellt werden (wie z.B. die Blöcke in Visuellen Programmiersprachen) und wie transparent die zugrundeliegenden Prozesse dadurch sind. Darüber hinaus geben Resnick et al. (2005) allgemeine Ratschläge für den Entwicklungsprozess und zur Nutzerbeteiligung, wie z.B. iteratives Vorgehen, partizipative Entwicklung und Langzeitevaluation.

De Roeck et al. (2012) stellen ähnliche Guidelines in einem Manifest für Entwickelnde und Designende von „internet-of-things creation platforms“ (De Roeck et al., 2012, S. 170) auf. Internet-of-things bezieht sich auf Anwendungen für intelligente, vernetzte Umgebungen, die auch als stofflich-digitale Artefakte eingeordnet werden können. An deren Erstellen und Gestaltung wollen De Roeck et al. (2012) Anwender_innen mit Hilfe von Plattformen mitwirken lassen. Zielgruppe sind explizit Amateur_innen, die im Kontext der Makerbewegung mit Arduino-Technologie arbeiten. Sie ist daher spezifischer als in den oben genannten Guidelines. Das Manifest stellt dreizehn Empfehlungen für Systeme zum Selbermachen von Internet-of-things-Artefakten auf. Eine solche Plattform sollte dazu motivieren, kreativ zu denken und gleichzeitig erfüllend sein sowie Flexibilität hinsichtlich der persönlichen Voraussetzungen und Vorlieben, insbesondere hinsichtlich der Programmierfähigkeiten, bieten. Sie sollte es ermöglichen, an fertige und unfertige Projekte oder Teilprojekte anderer anzuknüpfen, verschiedene Herangehensweisen unterstützen, Tinkering und Zusammenarbeit fördern und darüber hinaus noch fachliche Hilfe bieten. Ihre Ratschläge sehen De Roeck et al. (2012) als gewinnbringend für Plattformen an, die gemeinsames Konstruieren unterstützen.

Beide hier vorgestellten Guidelines beziehen sich auf ähnliche Zielgruppen und Tätigkeitsfelder wie diese Arbeit und geben daher relevante Hinweise für übergeordne-

te Rahmenbedingungen von Modellierungstools. Die genannten Zusammenstellungen von Guidelines spiegeln Werte wie Autonomie, Lernen, Berücksichtigung von Diversität und Kreativität wider. Es wird deutlich, dass Abwägungen hinsichtlich Flexibilität und Einfachheit getroffen werden müssen und sich hier ein Spannungsfeld auftut. Auf die engeren Ziele dieser Arbeit, nämlich was beim Unterstützen des Abbildens von Modellen zu beachten ist und welche Abbildungsformen sich eignen, geben die vorgestellten Guidelines jedoch keine Antwort.

Bei der Entwicklung der Anforderungen für Modellierungstools in Abschnitt 6.1.4 wird auf die vorgestellten Gestaltungsrichtlinien Bezug genommen.

5.5 Zusammenfassung und Schlussfolgerungen

In diesem Kapitel wurden verschiedene Methoden und Tools betrachtet, die als relevant für eher ‚unformale', für Laien verständliche Herangehensweisen an die Modellierung stofflich-digitaler Artefakte angesehen werden. Die Anwendungsfelder beschränkten sich dabei auf Modellierungstätigkeiten im Sinne des Dokumentierens von Modellierungsprozessen, als auch auf das Abbilden und Überführen unformal wahrgenommener Eigenschaften in formale Modelle.

Zunächst wurden in Makerkreisen gängige Tools zur Dokumentation und zum Austausch von Projekten betrachtet. Dazu zählen die How-tos, mit denen Modellierungsprozesse nachvollziehbar für andere abgebildet werden, wie auch Tools mit denen fertige Projekte dokumentiert werden. How-tos sind flexibel und prozessorientiert, jedoch sind hier Abbildungsmöglichkeiten für formale Gegebenheiten und Zusammenhänge nicht explizit integriert. Auch ist es nicht vorgesehen, nicht-lineares Vorgehen abzubilden.

Auch Storyboards beruhen auf dem Prinzip ‚Bild(er)-mit-Text', werden aber eher benutzt, um Interaktionsmöglichkeiten zu erfassen. Storyboarding wie auch Szenarien werden im Interaktionsdesign u.a. benutzt, um den insbesondere bei ubiquitären Systemen wichtigen Kontext abzubilden. Ferner sind sie allgemeinverständlich und können von allen am Prozess Beteiligten erstellt und verstanden werden. Einige Tools versuchen, Storyboarding mit formaler Modellierung zu verknüpfen.

Während Dokumentationstools wie How-tos oder auch Storyboards sehr flexible sind, bieten sie wenig Anhaltspunkte, wie ein stofflich-digitales Artefakt damit umfassend abgebildet werden kann oder wie Zusammenhänge zwischen konkreter Erscheinung und abstrakten Modellen explizit aufgezeigt werden können. Auch das Mitteilen von Intentionen wird nicht ausdrücklich gefördert, durch das narrative Format aber ermöglicht. Subjektkontexte lassen sich mit diesen Techniken und Tools abbilden, sofern darauf Wert gelegt wird.

Des Weiteren wurden Tools betrachtet, mit denen formale Eigenschaften stofflich-digitaler Artefakte ohne Fachwissen in Informatik modelliert werden können. Dabei

wurden zum einen Tools für Designer_innen, für Bildungszwecke und für angehende Maker_innen betrachtet, die Prinzipien von Übergängen zwischen verschiedenen Abstraktionsstufen mit ikonischen oder gegenständlichen Repräsentationen abstrakter Sachverhalte implementieren. Die Tools für Maker_innen beruhen auf visuellen ‚drag-and-drop'-Oberflächen, auf denen Hardware- oder Programm-Komponenten zusammengesetzt werden können und die automatisch durch das Softwaretool in Programmcode oder Schaltkreissymbolik übersetzt werden. Durch diesen spielerisch anmutenden, allgemeinverständlichen Ansatz können Programme oder Hardwarekonfigurationen ohne viel technisches Vorwissen zusammengesetzt werden. Durch ihre Oberfläche erscheinen sie eher unformal. Die Tools zum Erstellen von Programmen oder anderen formalen Modellen zeigen durch ihre Mehrebenen-Prinzipien und/oder das Verknüpfen verschieden abstrakter Repräsentationen Zusammenhänge zwischen Programmcode oder formalem Diagramm und vorliegendem, konkretem Artefakt auf. In Hinblick auf das übergeordnete Ziel dieser Arbeit, dem Abbilden von Modellen und ihren Entstehungsprozessen als Dokumentationen, fehlen den Programmier- und Hardwarekonfigurationstools Informationen über und Abbildungsmöglichkeiten für pragmatische und andere, im Artefakt manifestierte unformale Merkmale (z.B. wie das fertige Artefakt aussieht). Auch können Folgen von Modellen als Modellierungsprozesse nur eingeschränkt abgebildet werden.

Neben Tools und Methoden wurden in diesem Kapitel generelle Empfehlungen für die Gestaltung von Tools für Amateur_innen für kreatives Konstruieren stofflich-digitaler Artefakte betrachtet. Relevant für diese Arbeit erscheint es, sowohl Einfachheit als auch Flexibilität hinsichtlich bevorzugter Herangehensweisen, Arbeitsweisen und Vorwissen zu gewährleisten. Repräsentationen von Abstraktionen sollten bedacht gewählt werden.

Insgesamt bieten die in diesem Kapitel betrachteten Tools und Methoden weniger ein ganzheitliches, umfassendes Abbilden stofflich-digitaler Artefakte, sondern konzentrieren sich auf einzelne Aspekte von Modellierungsprozessen, wie oben bereits jeweils geschlussfolgert wurde. Es fehlen Tools für die genannte Zielgruppe, mit denen gleichzeitig unterschiedlich formale (Design- und Informatik-)Modelle, deren Übergänge und die Modellierungsprozesse abgebildet werden können.

Für diese Arbeit bleibt zu lösen, wie die relevanten Eigenschaften dieser unterschiedlichen Tools kombiniert werden können, so dass Artefakte ganzheitlich und zugleich als Modellierungsprozess abgebildet werden können, um ein eigenes Weitermodellieren oder sogar Weitergeben an andere zu ermöglichen. Als Voraussetzung, um Artefakte umfassend abbilden zu können, muss auch geklärt werden, welche Eigenschaften der Artefakte wesentlich sind und als welche Arten von Repräsentationen deren Modelle abgebildet werden können. Um weiteren Aufschluss über wünschenswerte Funktionen und Gestaltung eines integrativen Tools zu bekommen, werden im

folgenden Kapitel die bisherigen theoretischen Erkenntnisse miteinander in Verbindung gebracht und Anforderungen aus diesen abgeleitet.

Kapitel 6

Making als Modellbildung: Konzepte für Modellierungstools

Die bisherigen Kapitel beschäftigten sich mit dem Kontext ‚Making' und entwickelten einen theoretischen Ansatz, mit dem sich diese Tätigkeit als Modellierung und daraus hervorgehende Objekte als Modelle beschreiben lassen. Obwohl es sich bei stofflich-digitalen Artefakten um informatische Systeme handelt, wurde in dieser Arbeit nicht nur eine Perspektive informatischer Modellbildung eingenommen, sondern Modellierung auch als eine Designtätigkeit betrachtet, die ein unformaleres Herangehen erlaubt. Doch wie lässt sich diese Modellsicht auf das Selbermachen stofflich-digitaler Artefakte übertragen, und welche Konsequenzen lassen sich daraus ziehen, um das Abbilden der dabei entstehenden Modelle zu unterstützen?

Dieses Kapitel setzt die bisherigen theoretischen Erkenntnisse zueinander in Beziehung und zieht daraus Schlüsse für das Selbermachen stofflich-digitaler Artefakte – auch bezeichnet als ‚Making'. Um sie in die Praxis des Selbermachens zu übertragen, werden anschließend Modellierungswerkzeuge für das Entwickeln und Abbilden von Modellen konzipiert.

6.1 Making und Modellbildung

Der erste Teil dieser Arbeit begann mit einer Betrachtung des Phänomens Makerbewegung, die den Kontext für diese Arbeit liefert. Der Fokus wurde auf das Erstellen stofflich-digitaler Artefakte durch Amateur_innen gelegt. Anschließend wurden solche Artefakte in der Informatik verankert. Zur theoretischen Beschreibung ihres Entwicklungsprozesses und der dabei entstehenden Modelle wurde die Allgemeine Modelltheorie (AMT) herangezogen. Da das Erstellen stofflich-digitaler Artefakte nicht nur aus Informatik- und Ingenieursicht betrachtet werden kann und heute auch (Interaktions-)Designer_innen mit unformalerer Herangehensweise an solchen

Entwicklungen beteiligt sind, wurden Modellierungsprozesse auch aus der Perspektive des Designs dargestellt. In diesem Abschnitt werden diese theoretischen Ansätze auf das Making stofflich-digitaler Artefakte junger Amateur_innen übertragen.

6.1.1 Herangehensweisen an Modellierung

Die Allgemeine Modelltheorie (AMT) definiert Modelle als reduzierte Abbilder eines Originals, die zu einem bestimmten Zweck erstellt werden. Im Sinne der AMT können also Skizzen, Pläne, Ideen, Programmcode, Prototypen, Dokumente, Diagramme und viele mehr, die während des Entwicklungsprozesses für ein Produkt oder ein informatisches System erstellt werden, als Modelle angesehen werden. Folglich finden sich Modelle sowohl im Design als auch in der Informatik, und auf beiden Gebieten werden Modelle als zentraler Bestandteil des Entwicklungsprozesses eines Artefakts angesehen.

Bei der informatischen Modellbildung werden Modelle von Modellen gebildet. Anfangsmodelle informatischer Systeme können einer Fachwissenschaft entstammen, sind also dort als Modelle definiert worden. In der HCI werden Modelle zur Interaktion von Menschen mit Computern erstellt.[1] Solch eine Interaktion zwischen Mensch und Computer kann nur äußerst unzureichend als ein allgemeingültiges Modell generalisiert werden. Informatische Modellbildung beruht auf konzeptueller, mathematisch-formaler Modellierung. Modelle werden meist in Fachsprache symbolisch-abstrakt abgebildet. In der Informatik dienen Modellierungsmethoden und -techniken im Allgemeinen der Kommunikation und Dokumentation. Zu entwickelnde Systeme aus verschiedenen Perspektiven zu modellieren kann helfen, deren Komplexität zu bewältigen. In Form von (konkreten) Prototypen dienen Modelle der Verifikation von Anforderungen.

Modellbildung aus Designperspektive wurde als eher konstruktiv und generativ beschrieben und ist durch einen kreativen, teils intuitiven Prozess geleitet, indem sich die Designenden nicht auf formal ableitbare Kausalitäten berufen können. Modelle im Designkontext können als eine Reihe konkreter, meist materieller, Prototypen charakterisiert werden, die nicht unbedingt in direktem Zusammenhang zueinander (ent)stehen. Solche Prototypen im Design dienen der Reflexion, dem Entwickeln von Ideen und dem Sammeln von Erfahrungen mit der explorierten, im Modell manifestierten Umsetzung der Idee. Das Vorgehen im Design kann auch als bricoleurartiges Vorgehen angesehen werden, bei dem neue Ideen und Handlungen in der Situation entstehen. Modelle im Designprozess sind grundsätzlich eher als ikonisch und gegenständlich einzuordnen und benutzen weniger formale symbolische Fachsprachen. Der

[1] Was nicht ausschließt, dass auch Modelle anderer Fachgebiete implementiert sind, die zum Gegenstand der Interaktion werden. Ein Beispiel wäre hier die Installation „Der Schwarm“ (Hashagen, Büching & Schelhowe, 2009).

Einbezug von Designgebieten in die HCI bringt aber allgemeinverständlichere[2] Methoden und Prototyping-Techniken, z.B. Storyboarding, in Informatikkontexte ein.

Modelle spielen also hinsichtlich ihrer wesentlichen Merkmale und ihres Zweckes in Informatik und (Interaktions-)Design eine unterschiedliche Rolle.[3] Während die Informatik Werkzeuge und Herangehensweisen zum Modellieren stofflich-digitaler Artefakte liefert, die Kenntnisse formaler Sprachen erfordern, trägt das Design eher für Amateur_innen verständliche oder nachvollziehbare Modellierungsmethoden bei. Ein konkretes, designassoziiertes Herangehen kann auch genutzt werden, um formale Modellierung komplexer Systeme verständlicher dazustellen, indem Zusammenhänge und Übergänge zwischen konkreten Erscheinungsformen und den sich dahinter befindenden abstrakten Modellen offengelegt werden. Solche Ansätze werden beim Making stofflich-digitaler Artefakte relevant, wenn Anfänger_innen ihre Ideen formalisieren müssen oder die Formalisierungen anderer nachvollziehen wollen. Gängige Visuelle Programmierumgebungen (siehe Abschnitt 5.3.4) machen sich dies beispielsweise zu Nutze, indem sie konkret anmutende Interfaces verwenden und verschiedene Repräsentationsformen von Programmcode anbieten.

Selbermachen kann als ein kreativer Prozess aufgefasst werden, der sich informeller Informationsquellen bedient. Eigenmotivation und Freude an der Aktivität stehen im Vordergrund. Projektideen entspringen persönlichem Kontext, sind damit persönlich bedeutsam. Der Weg zu einem fertigen Artefakt muss dabei nicht zielgerichtet sein. Es kann auch im Sinne eines ‚tinkering'[4] entstehen, das von einer reflektierten Auseinandersetzung mit dem zur Verfügung stehenden Material gekennzeichnet ist. Auch mit welchen Mitteln eine Idee realisiert wird, ist meist zweitrangig. Damit ist Making – wie Design – eher als lösungsorientiert zu beschreiben und für die Selbermachenden reicht es aus, gerade genug zu wissen, um zu einer Lösung zu gelangen. Wie sehr von den jeweiligen Herangehensweisen dabei Gebrauch gemacht wird, hängt sicherlich von den eigenen Präferenzen und Vorerfahrungen ab. So kann angenommen werden, dass einige Menschen lieber in einem Bricoleur-Stil arbeiten, während andere eher planend vorgehen. Es soll hier davon ausgegangen werden, dass Selbermachende als junge Amateur_innen – und dies bestätigen auch generelle Beobachtungen in TechKreativ-Workshops – eher anhand konkreter Modelle ihre Ideen intuitiv entwickeln und modellieren und damit designtypisch vorgehen, da ihnen konzeptuelle formalisierende Modellierungsmethoden nicht vertraut sind. Diese Sichtweise soll nicht darüber hinweg täuschen, dass Designer_innen über spezielle, für Amateur_innen nicht intuitive, Methoden verfügen und dass auch Designdenken erlernt wird.

[2] Mit ‚allgemeinverständlich' ist gemeint, dass keine Fachsprache benutzt wird, sondern ikonische Bildmodelle oder umgangssprachliche textuelle Modelle.

[3] Was nicht ausschließen soll, dass individuelle Präferenzen und Unterschiede bestehen, insbesondere wenn die Tätigkeiten der Hard- oder Softwareentwicklung mit denen des Interaktionsdesigns verschwimmen, z.B. weil die Durchführenden auf beiden Tätigkeitsfeldern ausgebildet sind.

[4] Deutsch etwa: ‚herumbasteln', frickeln.

Ein interaktives, stofflich-digitales Artefakt wird in einem kreativen Prozess erfunden und entworfen, muss aber auch technisch implementiert werden, um Wirklichkeit zu werden. *Stofflich*-digitale Artefakte haben dabei die Besonderheit, dass auch ein stoffliches Produkt erstellt wird, dessen Konstruktion von vornherein ein unformaleres Herangehen erlaubt als die Umsetzung seiner digitalen Funktionalität. Somit sind Modellierungstätigkeiten involviert, die verschiedenes Vorgehen erfordern. Nicht alle Modelle können in kreativer Auseinandersetzung mit konkretem Material und Prototypen entstehen. Insbesondere die Programmierung erfordert eine Auseinandersetzung mit abstrakter Algorithmik und somit formalen Modellen. Wie in Abschnitt 2.5.4 ausgeführt wurde, wird als wichtig erachtet, dass Selbermachende ein Verständnis der programmierten Abstraktionen entwickeln können, um solche Artefakte ‚durchschauen', nachvollziehen und kompetent für sich nutzen zu können. Somit gilt es, Tools für junge Maker_innen zu entwickeln, die konkrete allgemeinverständliche Zugänge und formale Modelle verbinden und die es erlauben, sich auch mit den abstrakten Modellen auseinanderzusetzen und Zusammenhänge zu erkennen.

6.1.2 Making als Iterieren zwischen Modellen

Das Selbermachen stofflich-digitaler Artefakte kann als ein Arbeiten mit *models*-to-think-with aufgefasst werden, so wie im Konstruktionismus von „objects-to-think-with" (Papert, 1980, S. 150) oder beim Arbeiten mit Prototypen im Kontext des Interaktionsdesigns von „things to think with" (Brandt & Grunnet, 2000, S. 179) gesprochen wird. Im Kontext dieser Arbeit kann der reflektierende Prozess des fortführenden Modellierens anhand vorheriger Modelle gar als models-to-*model*-with bezeichnet werden, um den aktiven konstruierenden Charakter des Modellierens und die Vielfalt der involvierten Modelle (auch gedanklicher Art) zu unterstreichen. Durch die Reflexion an den sichtbar gewordenen Modellen wird das sich in Konstruktion befindende Artefakt (auch ein Modell) verändert und erweitert. Neue, revidierte Modelle, die sich dem Projektziel annähern, entstehen.

Doch wie äußern sich Modelle in Makerkontexten? Im Makerkontext finden wir Modelle als Projektdokumentationen, die im Internet veröffentlich werden. Diese Modelle dienen als Austauschmedien, bzw. werden erstellt, um Projekte weiterzugeben. Zum einen kann eine Projektdokumentation – z.B. ein How-to – als ein Modell des Entstehungsprozesses aufgefasst werden. Zum anderen enthält eine Projektdokumentation viele einzelne Elemente wie z.B. Programmcode, Bilder, Text, die jeweils als ein eigenes Modell angesehen werden können und verschiedene Perspektiven auf das Artefakt widerspiegeln. Und auch das Artefakt selbst und seine stofflichen Teile können als Modelle gelten. Diese Arbeit konzentriert sich auf beide Erscheinungsformen von Modellen – externe Repräsentationen von Modellierungsprozessen, als auch einzelne Abbilder verschiedener Aspekte eines Artefakts.

Im gesamten Modellierungsprozess entstehen Iterationen des Artefakts, ggf. ergänzt durch weitere Modelle mit unterschiedlichen Perspektiven. Ein typisches How-to bildet eine Kette von (nachträglich ausgewählten oder erstellten) Modellen ab, die als chronologische Schritte dargestellt werden. So kann das Modellieren, wie oben beschrieben, auch im Kontext des Making als Bilden von Modellketten aufgefasst werden. In einem Modellierungsprozess entstehen also verschiedene Modelle unterschiedlicher Granularitäten und aus verschiedenen Perspektiven, die miteinander auf verschiedene Weise in Bezug stehen. Es handelt sich daher mehr um eine vielfach verkettete Modellfolge mit Anhängseln oder auch um eine Art Modellkettennetz. Damit die vielfältigen Modelle Sinn ergeben und nachvollziehbar werden, müssen sie jedoch in Beziehung zueinander und zum Projektziel gebracht werden können. Im Falle von How-tos wird dies durch die chronologische, erzählende Struktur sowie textuelle Erklärungen erreicht. Allerdings haben How-tos eine lineare Struktur und erfordern viel Aufwand im Erstellen, so dass sie für das Festhalten von Modellen während eines Modellierungsprozesses als weniger geeignet angenommen werden. Deshalb sollen in dieser Arbeit neue Wege ausprobiert werden.

6.1.3 Modellperspektiven

Das Modellieren stofflich-digitaler Artefakte ist eine komplexe Tätigkeit, die durch diverse Modelle verschiedener Ausschnitte mit unterschiedlichen Perspektiven abgebildet werden kann. Aus Sicht der AMT reduziert sich ein Modell auf eine bestimmte Menge von Attributen des Originals. Im vorliegenden Fall ist ein Original, das den weiteren Konstruktionsprozess bestimmt, die Idee für das zu realisierende interaktive Artefakt. Im Laufe eines Entwicklungsprozesses entstehen verschiedene weitere Modelle, die jeweils nur eine bestimmte Menge von Attributen dieses Originals (oder eines weiteren Modells des Originals) zeigen. Durch so ein reduziertes Modell wird eine bestimmte Perspektive auf das zu modellierende Artefakt eingenommen. Diese Perspektive enthält die für sie relevanten Aspekte des Zielartefakts (bzw. dessen Vorstellung).

Hier soll den Perspektiven nachgegangen werden, aus denen Modelle selbstgemachter stofflich-digitaler Artefakte erstellt werden (können). Diese sind in Hinblick auf die Entwicklung von Modellierungstools (siehe Abschnitt 6.5) von Interesse, die angehende Maker_innen beim Nachvollziehen und Abbilden ihrer Projekte unterstützen sollen. Außerdem dienen sie als Kategorien und Dimensionen für eine Analyse von Modellen, die in Abschnitt 7.3 getätigt wird, um herauszufinden, welche Modellarten Amateur_innen zum Abbilden bestimmter Artefakteigenschaften nutzen.

Stofflich-digitale Artefakte sind komplexe, vielschichtige Systeme. An solchen Projekten sind – aus professioneller Sicht – verschiedene Fachrichtungen beteiligt, deren Sichtweisen eingenommen werden können und die verschiedene Modelle einbringen, so wie es Buur und Andreasen (1989) im Mechatronik-Kontext beschreiben (siehe Ab-

schnitt 4.5). Aus unterschiedlichen fachlichen Perspektiven werden Modellfolgen mit Modellen erstellt, die verschiedene Detailliertheitsgrade und Aspekte des zu Modellierenden abbilden. Im Falle von Mechatroniksystemen wurden die Teilsysteme Software, Elektronik und Mechanik genannt, in denen jeweils Modelle erstellt werden. Das Einnehmen verschiedener Sichtweisen und entsprechende Modelle eines Systems können helfen, dessen Komplexität zu bewältigen (siehe z.B. Abschnitt 3.2.4). Bei Makerprojekten mögen Modelle nicht eindeutig nach Fachrichtungen trennbar sein, dennoch besteht ein stofflich-digitales Artefakt aus verschiedenen Komponenten, die mehr oder weniger getrennt voneinander modelliert werden (können). Doch aus welchen Teilsystemen besteht ein selbstgemachtes stofflich-digitales Artefakt? Welche Perspektiven resultieren daraus, aus denen Modelle für diese Komponenten bzw. Aspekte erstellt werden können oder müssen? Um dem nachzugehen, soll exemplarisch ein Artefakt, das in einem TechKreativ-Workshop entstanden ist, genauer betrachtet werden.

Bei dem Artefakt handelt es sich um einen ‚rollenden Kopf' (Abbildung 6.1), der „Disco-man" genannt wurde. Das Projekt entstand in einem Workshop[5] im Jahre 2012. Dieses Objekt wurde gewählt, weil es komplett erhalten und funktionstüchtig war und eine kurze Dokumentation von Funktionsweise und Programm (erstellt von den Workshopdurchführenden zu Evaluationszwecken) vorlag.

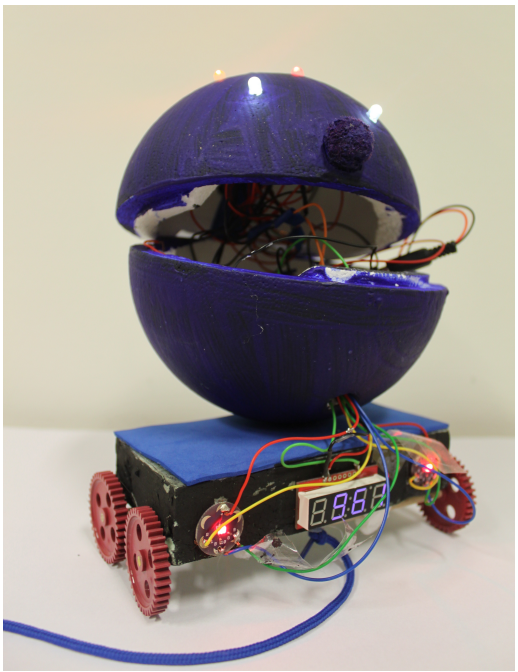


Abbildung 6.1: Artefakt ‚rollender Kopf' bzw. „Disco-man" (Foto: Jennifer Boldt)

Äußerlich handelt es sich um einen Kopf auf einem Fahrgestell mit Rädern. Der Korpus des Objekts besteht im Wesentlichen aus einer Styroporkugel, die lilafarben

[5]Dieser wurde im Rahmen des Forschungsprojekts informAttraktiv (`http://www.dimeb.de/informattraktiv`, aufgerufen am 14.05.2014) veranstaltet (siehe auch Dittert, Wajda und Schelhowe (2015)).

angemalt wurde und eine Nase aus bemaltem Kork hat. Die Kugel besteht aus zwei Hälften, die geöffnet werden können. Sie ist auf einem schwarz bemalten Fahrgestell aus Styrodor montiert, das zwei Achsen mit je zwei roten Plastikrädern besitzt. An diesem Unterteil ist eine Schnur befestigt. Auf dem Fahrgestell sind zwei RGB-LEDs, ein Lichtsensor und ein numerisches Display angebracht, von denen Kabel ins Innere der Kugel führen. Auf der Kugel sind die Leuchtkolben von LEDs zu sehen, die u.a. als Augen dienen. Öffnet man die Styroporkugel, so finden sich weitere Hardwarekomponenten. Diese umfassen ein Arduino-Board, an dessen Ein- und Ausgänge Kabel angeschlossen sind, die zu den zuvor genannten Aktuatoren und Sensoren führen. Außerdem ist ein Batterieanschluss zur Stromversorgung am Board vorhanden. Wird das Board mit Strom versorgt, so wird der bisher statische Kopf zu einem interaktiven Artefakt. Wird es bewegt (z.B. gezogen) und das Umgebungslicht gedimmt, so leuchten die LEDs, die als Augen am Kopf und am Fahrgestell angebracht sind, in verschiedenen Mustern und das Display zeigt Ziffern an. Dieses Verhalten wird durch die Programmierung des Boards erreicht, die festlegt, wie die LEDs auf die Sensorwerte reagieren und dass das Display die Werte des Lichtsensors anzeigt.

Doch welche Teilsysteme wurden für den ‚rollenden Kopf' konstruiert und aus welchen Perspektiven wurden dafür Modelle erstellt? Das Artefakt besteht hauptsächlich aus einem stofflichen Körper, der allem Anschein nach mittels unformaler Modellierung und konkreten Materialien gebastelt wurde. Um diesen interaktiv, d.h. zum digitalen Artefakt, werden zu lassen, wurde das dem Artefaktkorpus zugedachte Verhalten programmiert. Dafür musste dieses formalisiert und als ein Algorithmus beschrieben werden. Die Implementierung des Verhaltens erforderte auch, es auf Sensoren und Aktuatoren zu reduzieren. Diese wurden mit einem Arduino-Board als Computer verbunden und in den Artefaktkorpus eingebaut. So machen sie den stofflichen Körper des Artefakts zum Interface, durch das das Programm wirken kann. Anhand dieser Betrachtung können als wesentliche Tätigkeiten bzw. Teile des Systems die Konstruktion des Artefaktkorpus, die Konfiguration der Hardware sowie das Codieren des Programms gelten. Es kann angenommen werden, dass dafür Modelle erstellt wurden, die jeweils eine Perspektive auf diese Teilaspekte einnehmen: Zum Beispiel eine Skizze des Körpers, ein Prototyp des Hardwareaufbaus, ein Programm.

Diese identifizierten Aspekte beruhen auf einer äußerlichen Betrachtung des Artefakts. Für diese Arbeit ist aber auch relevant, wie Modelle dargestellt werden können, die es erlauben, Funktionsweise und Wirken eines Artefakts nachzuvollziehen. Deshalb ist von Interesse, welche weiteren Modellperspektiven in dieser Hinsicht sinnvoll sein können.

Im Modellierungsprozess wird ein der Wirklichkeit entstammendes Modell in ein abstraktes Modell formalisiert (siehe Abschnitt 3). Als laufendes Programm wirkt es durch das stoffliche Interface des Artefakts und kann interaktiv erlebt werden. Das Verhalten des Artefakts wird so für Außenstehende konkret wahrgenommen. Die Zu-

sammenhänge zwischen abstraktem Programm und konkreter Äußerung können und sollten beim Konstruieren des Artefakts nachvollziehbar sein, so dass Konstruierende etwas über das Wesen Digitaler Medien erfahren (siehe Kapitel 2). Sind der Zusammenhang zwischen abstraktem Programm und konkretem Verhalten eines Artefakts anhand der dokumentierten Modelle nachvollziehbar, so kann es vermutlich auch einfacher den eigenen Bedürfnissen angepasst werden. Daher erscheint es sinnvoll, Modelle des Verhaltens eines Artefakts (auch) aus Perspektiven zu erstellen und zu dokumentieren, die zwischen formaler und unformaler Seite unterscheiden und/oder bewusst Übergänge darlegen. Deshalb sollen die drei Modellierungsperspektiven Artefaktkorpusgestalt, Hardwarekonfiguration (abgekürzt ‚Hardware'[6]) und Programmierung (abgekürzt ‚Programm') um eine Perspektive auf das interaktive Verhalten erweitert werden, die das Wirken des Artefakts aus einer gegenständlich-konkreten, unformalen Sicht zeigt.

Die Perspektive des interaktiven Verhaltens stellt ein konkretes unformales Gegenstück zur Programmierung und der resultierenden Hardwarekonfiguration des Artefakts dar. Somit handelt es sich um einen dynamischen, zeitgebundenen Aspekt, der den Korpus des Artefakts nicht unmittelbar einschließt. Im vorliegenden Fall – und generell in TechKreativ-Workshops – wurde der Artefaktkorpus sehr wahrscheinlich unformal modelliert. Deshalb wird es für den vorliegenden Fall nicht als notwendig erachtet, den Artefaktkorpus um eine zusätzliche unformale Perspektive zu ergänzen, da dieser bereits unformal ist und auch bleibt.[7] Je nach Art eines Artefakts sind weitere Perspektiven denkbar, zum Beispiel für Modelle, die eine Mechanik abbilden.[8]

Zusammenfassend sollen das interaktive Verhalten, die Programmierung, die Hardwarekonfiguration und der Artefaktkorpus als wesentliche Aspekte gelten, die ein selbstgemachtes stofflich-digitales Artefakt kennzeichnen. Sie stehen für Perspektiven, aus denen ein Artefakt modelliert werden kann und die als hilfreich angesehen werden, um ein Projekt mit reduzierter Komplexität zu modellieren und nachvollziehbar zu dokumentieren. Diese Modellperspektiven sind in Abbildung 6.2 dargestellt.

Passende Modellarten

Die identifizierten Modellperspektiven erfordern bestimmte Modelltypen, die in der Lage sind, die jeweiligen Spezifika abzubilden. Für die Gestaltung und den Entwurf der körperlichen Form des Artefakts und seines angedachten Verhaltens sind allgemeinverständliche und nicht-formale Darstellungsmöglichkeiten denkbar, z.B. eine

[6] Damit sind elektronische Bauteile wie das Mikrocontroller-Board, Aktuatoren, Sensoren u.ä. gemeint.

[7] Mit dem Aufkommen von Fabrikationstechnologien kann sich dies ändern. Dann muss auch der Artefaktkorpus formalisiert werden, um ihn z.B. mit einem 3D-Drucker drucken zu können. Zum Zeitpunkt der Erhebungen wurden solche Maschinen noch nicht in TechKreativ-Workshops zusammen mit Physical Computing eingesetzt.

[8] Dies kann z.B. erforderlich sein, wenn sich ein Hebel bewegen soll, oder um Statik zu berücksichtigen, ist aber i.d.R. kein essentieller Teil von TechKreativ-Projekten und wird deshalb hier ebenfalls vernachlässigt.

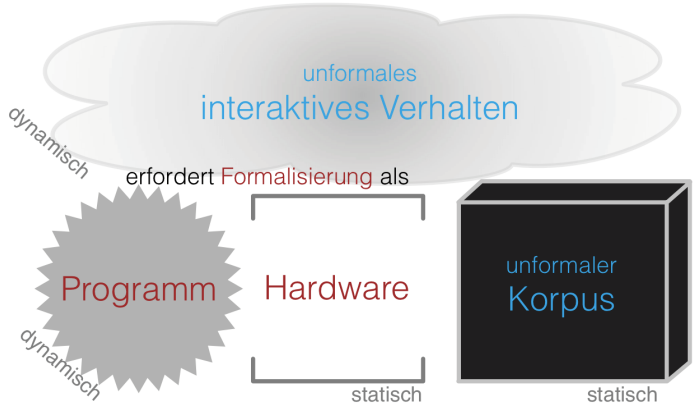


Abbildung 6.2: Modellperspektiven auf ein selbstgemachtes stofflich-digitales Artefakt

Skizze, ein Mock-up oder ein Storyboard. Die Programmierung erfordert zwingend eine Transformation in ein formales Modell (i.d.R. Programmcode), und auch die Hardwarekonfiguration ist formalsprachlich beschreibbar, z.B. als Schaltplan. Doch es gibt noch eine weitere Unterscheidung der gewählten Perspektiven zu bedenken, die die Wahl geeigneter Modellarten einschränkt: Modelle des interaktiven Verhaltens und der Programmierung beschreiben dynamisch-interaktive Eigenschaften des Artefakts. Demzufolge werden dynamische Modelle benötigt oder z.B. statische diagrammatische Modelle, die in der Lage sind, diese Dynamik abzubilden. So kann das zunächst unformal und umgangssprachlich modellierte Verhalten des Artefakts z.B. als Storyboard festgehalten werden. Im Laufe der Entwicklung muss es genauer spezifiziert und als Programmcode formalisiert werden. Im professionellen Kontext gibt es dafür auch diagrammatische Modellierungssprachen. Der unformale Artefaktkorpus und die formal beschreibbare Hardwarekonfiguration sind dagegen statischer Natur und lassen sich mit statischen Modellen modellieren. Dies können Skizzen, Mock-ups oder Schaltbilder sein. Damit unterscheiden sich die vier Perspektiven nicht nur durch die Eigenschaften formal und unformal. Auch haben sie unterschiedliche dynamische und statische Eigenschaften, die passende Modellarten für ihre Darstellung erfordern.

Pragmatische Merkmale beim Modellieren

Unser Beispielobjekt, der ‚rollende Kopf', verrät nichts über die Motivation, die ihm zugrunde liegt. Auch erfahren wir nicht, wozu er erstellt wurde, und ob das Artefakt, das wir wahrnehmen, dem entspricht, was seine Erfinder_innen in ihm sehen. So lässt sein Name „Disco-man" nur erahnen, für welchen Kontext er angedacht war.

Wie in Kapitel 3 beschrieben wurde, entspricht die Idee für das Artefakt nicht immer dem tatsächlich modellierten Objekt, bzw. wird von unterschiedlichen Modellsubjekten verschieden wahrgenommen und wirkt sich anders als geplant auf deren Kontexte aus. Aus Sicht der AMT sind ‚subjektive' Faktoren in Form der pragmatischen

Merkmale ein wesentliches Charakteristikum, das ein Modell definiert. Intentionen, Modellsubjekte, Zeit und Zweck spielen bei der Herstellung eine wichtige Rolle und können sich auf die Nutzung auswirken. Wenn Modelle stofflich-digitaler Artefakte dokumentiert und ausgetauscht werden, so reicht es nicht aus, z.B. nur das Programm und den Hardwareaufbau und andere äußere Eigenschaften zu dokumentieren. Der Zeitpunkt der Umsetzung legt zum Beispiel fest, welche technischen Möglichkeiten und Rahmenbedingungen zur Verfügung standen (z.B. die Softwareversion und Variante des Arduino-Boards). Intentionen und Projektziel sind wesentliche Informationen, um Entscheidungen nachvollziehen zu können. Auch Informationen über das Subjekt, z.B. seine Vorkenntnisse, können relevant sein und beeinflussen das Modell. Über das übergeordnete Ziel eines Artefakts hinaus hat jedes Modell, das im Laufe des Konstruktionsprozesses erstellt wird, auch einen eigenen Zweck. Zum Beispiel dient eine Skizze dazu, das Projekt zu veranschaulichen und die Hardwarekonfiguration zu planen, oder ein Foto dazu, den aktuellen Stand festzuhalten, um später ein How-to zu erstellen.

Pragmatische Merkmale, die Auskunft über Intentionen, den Modellkontext und die Modellsubjekte geben, sind also ein essentieller Bestandteil von Modellen. Sie sind von Bedeutung, um anderen ein möglichst vollständiges Bild des Projekts zu vermitteln, das nicht allein aus dem Artefakt ersichtlich ist. Daher sollen pragmatischen Merkmale, die den Modellen des Artefakts zugrunde liegen, als eine weitere relevante, übergeordnete Perspektive gelten. Diese fünfte Perspektive wurde als Schatten in Abbildung 6.3 integriert, da sie alle anderen Modellperspektiven auch betrifft.

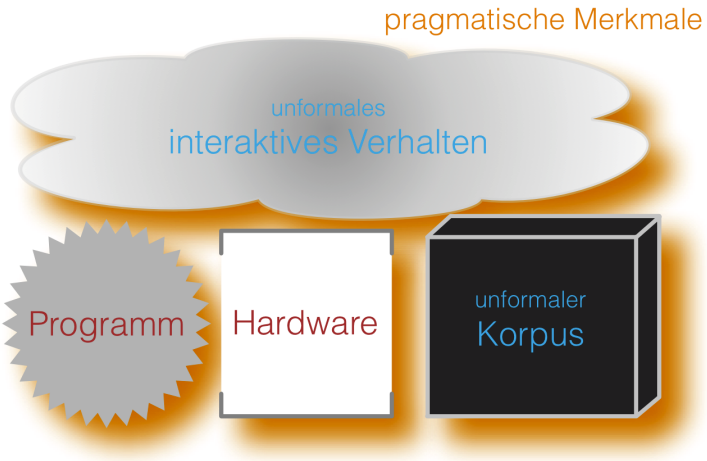


Abbildung 6.3: Ergänzung der Modellperspektiven um pragmatische Merkmale

Zusammenhang der Perspektiven

Die aufgestellten Perspektiven können nicht als isoliert und zeitlich aufeinander aufbauend betrachtet werden. Making wurde oben als ein iterierender Modellierungspro-

zess bezeichnet. Sobald die Teilnehmenden eines TechKreativ-Workshops das Verhalten für ihr Artefakt entworfen haben, können sie die dafür notwendige Hardwarekonfiguration entwickeln. Diese und auch die geplanten Interaktionsabläufe können wiederum die Gestalt des Artefaktkorpus bestimmen. So musste dieser beim ‚rollenden Kopf' groß genug sein, um die Hardware darin verstecken zu können. Diese Aspekte können wiederum die Programmierung beeinflussen, z.B. weil die Sensoren anders als erwartet reagieren oder sich die Gestalt des Korpus auf das Interaktionsverhalten der Nutzenden auswirkt. Auch die Programmiersprache schränkt die vorhandenen Möglichkeiten ein und kann Änderungen der anderen Komponenten und der Projektidee erfordern. Neu entdeckte Möglichkeiten und Beschränkungen können sich auf die Ausgestaltung der übrigen Aspekte auswirken, so dass sich das als Vorbild dienende (ggf. gedankliche) Konzept laufend wandelt.

Demzufolge gibt es kein generelles allgemeingültiges Vorbildmodell, das am Anfang des Prozesses steht und konsequent umgesetzt wird. Vielmehr entwickelt sich die Idee, das Vorbild- oder Zielmodell, im Laufe des Prozesses immer wieder neu. Durch Auseinandersetzung mit dem Material und dessen Gegebenheiten eröffnen sich ggf. neue Möglichkeiten, alte werden verworfen. Sofern keine oder nur vage Anforderungen von außen vorliegen, können sich beim kreativen Arbeiten Idee und Ziel weiterhin ändern. Vorbildmodelle, die die Idee repräsentieren, entstehen also kontinuierlich und manifestieren sich im Laufe des Prozesses im Artefakt, an dem gearbeitet wird. Modelle, die die verschiedenen Perspektiven einnehmen, beeinflussen sich damit ständig gegenseitig und führen zu Änderungen. Abbildung 6.2 zeigt die identifizierten Modellperspektiven, jedoch nicht *wie* und in welcher Reihenfolge aus diesen modelliert wird.

6.1.4 Anforderungen an Modellierungstools und -methoden

Aus den theoretischen Überlegungen werden Anforderungen für Modellierungsmethoden und -tools abgeleitet, die Amateur_innen beim Abbilden von stofflich-digitalen Artefakten bzw. deren Modellen unterstützen sollen:

1. **Allgemeinverständlichkeit:** Tools oder Methoden sollen einen unformalen, allgemeinverständlichen Zugang für junge Amateur_innen zur Modellierung stofflich-digitaler Artefakte bieten, ohne Fachwissen vorauszusetzen. Es wird angenommen, dass am ehesten etwas über das Wesen Digitaler Medien erfahren werden kann, wenn die dort dokumentierten Artefakte tatsächlich realisiert werden können (siehe Abschnitt 2.5.4 und 5.1.3). Deshalb sollen Modellierungstools dabei helfen, abstrakte Prozesse verständlich darzustellen. Dabei sollen formale, informatische Sachverhalte nicht versteckt werden, sondern so abgebildet werden können, dass sie nachvollziehbar sind (siehe Abschnitt 2.5.4).

2. **Persönlicher Kontext:** Die Ideen, die Maker_innen umsetzen, entstammen ihrem persönlichen Kontext und sind entsprechend motiviert (siehe Kapitel 2). Soziale, persönliche Kontexte spielen auch eine kritische Rolle beim Modellieren, wenn informatische Modelle in anderen Kontexten wirken, als sie entstammen. Das Dekontextualisieren von Modellen trägt zu Zielkonflikten der Formalisierung bei (siehe Abschnitt 3.3) und kann das Nachvollziehen von Projekten erschweren. Modellierungstools sollen aus diesen Gründen Möglichkeiten anbieten, persönliche Kontexte der Modellierenden und der Artefakte als pragmatische Merkmale abzubilden.

3. **Flexibilität:** Maker_innen gelten in dieser Arbeit als Amateur_innen ohne großes fachlich-methodisches Vorwissen, denen deshalb konkrete Zugänge helfen können. Auch haben Menschen verschiedene Vorlieben, wie sie beim Modellieren von etwas vorgehen (siehe Konzept der Planners und Bricoleurs in Abschnitt 4.3.2). Deshalb sind verschiedene Herangehensweisen an Modellierung zu unterstützen (siehe auch Guidelines in Abschnitt 5.4). Ein Modellierungstool soll flexibel hinsichtlich verschiedenen Modelldarstellungen sein und es erlauben, verschiedene Modelltypen einzubinden – auch solche, die analog oder mit externen digitalen Tools erstellt werden. Auch sollen Abbildungsmöglichkeiten für ungewöhnliche Modelle bestehen, z.B. um körperliches Vormachen zu Dokumentieren.

4. **Praxisintegration:** Dokumentieren ist aufwändig und unbeliebt. Die Tätigkeit des selbstmotivierten Selbermachens steht für die Schaffenden im Vordergrund. Sie sollen diese nicht grundlegend unterbrochen müssen, um Modelle zu dokumentieren (siehe Kapitel 2). Nutzende sollen ihre gewohnten Praktiken beibehalten können (siehe Guidelines von Resnick et al. (2005) in Abschnitt 5.4). Demzufolge soll sich ein Modellierungstool in die Praxis des ‚Making' integrieren. Es soll den Prozess möglichst nicht unterbrechen und nicht von den aktuellen Konstruktionstätigkeiten ablenken.

5. **Einfachheit:** Die wesentlichen Funktionen eines Modellierungstools sollen einfach zu erreichen und zu überblicken sein. So können sich die Nutzenden auf die wesentlichen Aspekte des Modellierens konzentrieren. Damit wird auch die Praxisintegration berücksichtig, indem der Aufwand und die notwendige Unterbrechung, um zu dokumentieren, minimiert werden. Entsprechend wird Einfachheit auch in den betrachteten Guidelines von Resnick et al. (2005) zur Verbesserung der User Experience gefordert (siehe Abschnitt 5.4).

6. **Perspektivenvielfalt:** Artefakte, die in Physical-Computing-Projekten entstehen, sind komplex. Sie haben z.B. eine individuelle stoffliche Form und Gestalt, ein programmiertes Verhalten und ein technisches ‚Innenleben'. Darüber hinaus besitzen sie pragmatische Merkmale, wie die dem Projekt zugrunde liegen-

den Intentionen. Um Dritten ein umfassendes Bild der Artefakte zu übermitteln, ist es daher notwendig, alle diese Eigenschaften abzubilden. Folglich sollen dynamisch-interaktive, statische, formale und unformale Eigenschaften des Artefakts sowie pragmatische Merkmale, z.B. Intentionen und Zweck des Projekts, in einer Dokumentation enthalten sein (siehe Abschnitt 6.1.3), aber auch durch verschiedene Arten von Repräsentationen abgebildet werden können. Das Berücksichtigen dieser Perspektivenvielfalt ist Voraussetzung, dass Artefakte und die damit verbundenen Intentionen vollständig genug abgebildet und nachvollziehbar für andere Maker_innen werden, die sich räumlich und zeitlich getrennt mit der Dokumentation eines Artefakts befassen (siehe Kapitel 2). Gleichzeitig kann die Komplexität des Artefakts durch verschiedene Perspektiven auf Teilsysteme reduziert werden und ggf. zu einem besseren Verständnis des Systems und der dahinter liegenden Konzepte beitragen (siehe Abschnitt 3.2.4). So kann ein Artefakt einerseits aus der Perspektive auf die in ihm steckenden abstrakten Modelle betrachtet und dokumentiert werden und andererseits aus der konkreten, gegenständlichen, erlebten Perspektive.

Folglich soll ein Modellierungstool das Abbilden von unterschiedlichen Modellen, die das Artefakt aus vielfältigen Perspektiven darstellen, ermöglichen.

7. **Perspektivtrennung:** Um verschiedene Perspektiven unterscheiden und einnehmen zu können, soll ein Modellierungstool es ermöglichen, Modelle getrennt nach Perspektiven zu erstellen bzw. zu dokumentieren. Auch für Betrachtende soll die jeweilige Modellperspektive ersichtlich sein. Damit knüpft diese Anforderung an die zuvor aufgestellte Perspektivenvielfalt an. Mögliche geeignete Perspektiven wurden oben aufgestellt (siehe Abschnitt 6.1.3).

8. **Modellnetze:** Sowohl informatische Modellbildung als auch Modellierung im Design kann als ein Netz von (oft chronologisch) aufeinanderfolgenden Modellen beschrieben werden, die aber auch Abzweigungen und ‚Anhängsel' enthalten (siehe Abschnitte 3.2.2 und 6.1.2). Ein Modellierungstool soll das Bilden von Modellketten oder -netzen, also Folgen von zueinander in Beziehung stehenden Modellen im Sinne eines Modellierens von Modellen, unterstützen.

9. **Modellschnittstellen:** Das Vernetzen von Modellfolgen über verschiedene Tätigkeitsfelder hinweg setzt voraus, dass Schnittstellen oder Übergänge zwischen Modellen dargestellt werden können. Trotz Perspektivtrennung sollen verschiedene Modellarten bzw. -folgen zueinander in Beziehung gebracht werden, so dass das zu modellierende System als Ganzes betrachtet werden kann. Dies wird u.a. mit Abhängigkeiten der Teilsysteme voneinander begründet (siehe Abschnitt 4.5). So können Modellnetze über Fachgebiete hinweg entstehen, und nicht nur isolierte Modellketten (siehe Abschnitt 3.2.4). Doch nicht nur über Tätigkeitsfelder hinweg sind Modellschnittstellen zu berücksichtigen. Auch um Modelle zu

vernetzen, die verschiedene Stufen der Abstraktion desselben Aspekts darlegen, sind Übergänge notwendig. Sie helfen beim Nachvollziehen und Verstehen der im Artefakt implementierten Konzepte (siehe Abschnitt 4.5.2).

Um diese Anforderungen zu überprüfen und den generellen Fragen dieser Arbeit nachzugehen, werden Tools und Methoden entwickelt. Sie verkörpern Konzepte dieser Arbeit und können zum Modellieren stofflich-digitaler Artefakte genutzt werden.

Einige der Anforderungen sind schwer miteinander vereinbar. So steht Einfachheit u. U. der Flexibilität entgegen. Das Einnehmen getrennter Perspektiven ist nur bedingt mit dem Abbilden von Schnittstellen vereinbar. Deshalb wurden mehrere Modellierungstools und -methoden entwickelt (bzw. weiterentwickelt) und erprobt, die sich auf bestimmte Anforderungen konzentrieren. Anhand von Evaluationsergebnissen werden später (in Abschnitt 7.5.5) die oben genannten Anforderungen revidiert.

6.2 Vorgehen und Entwicklungsmethodik

Bevor die Entwicklungen vorgestellt werden, erfolgt ein Überblick über Methodik und Vorgehen, um sie in den Gesamtkontext der Arbeit einordnen zu können.

6.2.1 Überblick

Für diese Arbeit wurden zwei Modellierungstools (weiter-)entwickelt: die bereits bekannte VPL Amici mit dem Schwerpunkt der Programmierung und die neue Webplattform *MoLab* mit dem Schwerpunkt des Dokumentierens von Modellierungsprozessen. Auch wurden zwei ‚analoge' Modellierungsmethoden konzipiert und erprobt: ein relativ unformales Skizzieren von Schaltkreisen und das Vorspielen von Szenarien.

Amici wird und wurde bereits seit ca. 2007 regelmäßig in TechKreativ-Workshops als Programmierumgebung eingesetzt und weiterentwickelt. MoLab dagegen wurde nur in drei Workshops im Rahmen dieser Arbeit eingesetzt, von denen der erste als Fallstudie evaluiert wurde. Parallel dazu wurden Modelle analysiert, die mit diesen beiden Tools, dem VirtualLab und analogen Modellierungsmethoden in diversen TechKreativ-Workshops entstanden sind (siehe Abschnitt 7.3). Die Ergebnisse der Untersuchungen fließen später in das Tool Amici+ ein (siehe Abschnitt 7.5.7). Abbildung 6.4 zeigt die wesentlichen Schritte des Vorgehens.

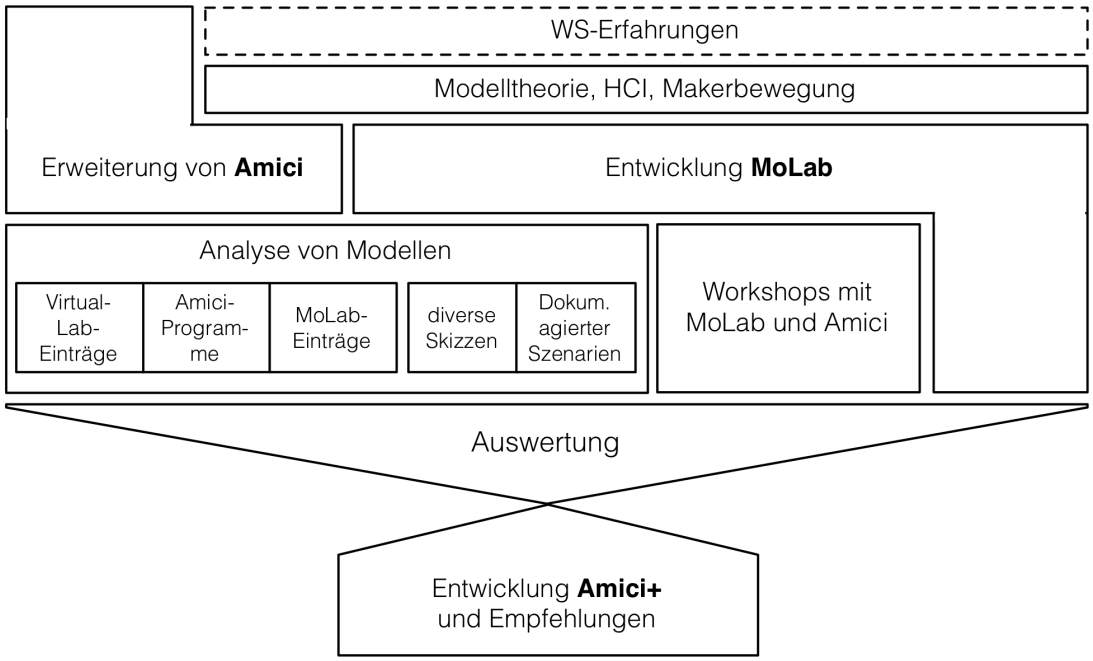


Abbildung 6.4: Vorgehen und Einordnung der Entwicklungsschritte

6.2.2 Vorgehen iterativer Entwicklung

Die Tools Amici und MoLab wurden in einem iterativen Vorgehen entwickelt. Die Erweiterungen von Amici, die für diese Arbeit relevant sind, wurden ab 2010 implementiert und seitdem in Workshops eingesetzt. Sie werden unten ausführlich beschrieben. Anhand schriftlich festgehaltener Beobachtungen von Workshoptutor_innen erfolgten außerdem regelmäßig kleinere Anpassungen, die sich in erster Linie an der Funktion als Programmierumgebung orientierten und der Fehlerbehebung dienten.[9] Eine Auswertung hinsichtlich modelltheoretischer Überlegungen und der Eignung als *Modellierungs*tool erfolgte erst im Rahmen dieser Arbeit.

MoLab wurde zunächst in einem mehrwöchigen Workshop mit sieben Teilnehmenden angewandt und währenddessen iterativ angepasst (siehe Abschnitt 7.4). Anpassungen erfolgten aufgrund von Beobachtungen (Protokollen) und einem auswertenden Gespräch. Daraus wurden Verbesserungen abgeleitet und implementiert. Diesen Änderungen wurde bei den nachfolgenden Terminen besondere Beachtung seitens der Workshopdurchführenden geschenkt, um die Entscheidungen zu überprüfen. Anschließend wurde das überarbeitete MoLab (2.0) noch in zwei mehrtägigen Workshops eingesetzt, aber nicht erneut evaluiert.

[9] Insgesamt flossen in die Weiterentwicklung von Amici Erfahrungen ein, die auf mindestens 25 TechKreativ-Workshops der AG dimeb mit schätzungsweise über 300 Teilnehmenden seit ca. 2007 beruhen. Die Autorin der Arbeit war davon an ca. zehn Workshops beteiligt.

Das Vorgehen bei der Entwicklung von Amici und MoLab ist an die Methode des design-based research angelehnt. „Design-based research“ (Obrenović, 2011, S. 56) oder „design studies“ (Confrey, 2006, S. 135) hat sich zunächst in den Lernwissenschaften aus methodischen Einflüssen von Piaget, Vygotsky und Dewey entwickelt (vgl. Confrey, 2006, S.136). Die Methode geht davon aus, dass eine Lernumgebung komplex und konditional ist und nicht deterministisch. Deshalb werden Entscheidungen in der Situation getroffen und der Untersuchungsgegenstand entsprechend angepasst (Confrey, 2006). Mittlerweile hat der Ansatz auch in der HCI Anklang gefunden (z.B. Obrenović, 2011). Er ergänzt etablierte empirisch-analytische Methoden, indem in realen Situationen geforscht wird und unmittelbar Anpassungen erfolgen können. So können auch theoretisch nicht vollkommen durchdrungene, auf implizitem situationsbezogenem Wissen beruhende Entwicklungen und Entscheidungen einbezogen werden (Obrenović, 2011). Der Nachteil von design-based research liegt in der erschwerten Generalisierbarkeit und in der über die jeweilige Studie hinausgehenden Validierung von Ergebnissen (Confrey, 2006).

Da design-based research die Situationsbezogenheit berücksichtigt, ist der Ansatz für die Evaluation und Weiterentwicklung von Digitalen Medien in Kontexten mit zu erwartender Eigendynamik interessant – so wie im vorliegenden Fall. Auch wurde dieser Ansatz gewählt, weil er die Problematik des Rückbezugs und des sich ändernden Kontexts Digitaler Medien berücksichtigt. Diese führen dazu, dass die Subjektwelt sich durch die Anwendung des Digitalen Mediums ändert und somit eine objektive Auswertung der Benutzung nicht möglich ist.

Im folgenden werden die (Weiter-)Entwicklungen der Methoden und Tools beschrieben und begründet.

6.3 Analoge Modellierungsmethoden

Um allgemeinverständliche, nicht-formale Modellierungsansätze zu explorieren, wurden Methoden entwickelt und erprobt, die ohne Softwareanwendung auskommen. Sie erlaubten es, eher ungewöhnliche Ansätze der Modellierung auszuprobieren und sind inspiriert von ähnlichen Techniken aus dem Participatory Design wie paper prototyping (z.B. Sanders et al., 2010) und performances (z.B. Macaulay et al., 2006). Da diese ‚analogen‘ Methoden bewusst nur in einzelnen Phasen eines Workshops eingesetzt wurden, bilden sie keine *Modellnetze* und nehmen nur eine oder wenige Perspektiven auf ein Artefakt ein.

6.3.1 Stromkreisskizzen für Smart Textiles

Skizzen sind ein beliebtes Mittel, um Modelle anzufertigen. Für einen Workshop zum Thema ‚smart fashion‘ im Juni 2008 wurden dazu spezielle Vorlagen konzipiert. Diese zeigten Umrisse von Kleidungsstücken – jeweils von vorne und von hinten gesehen

(siehe Abbildung 6.5). Die Aufgabe für die Teilnehmenden war, den Verlauf des leitfähigen Garns, das als Stromleitung dient, dort einzuzeichnen. Es sollte also vor der Konstruktionsphase, bevor das leitfähige Garn angenäht wurde, eine Art Schaltbild erstellt werden. Zum Einzeichnen standen Buntstifte zur Verfügung. Ziel war es, die konkrete Artefaktgestalt mit der Hardwareskizze zu verbinden und ein wenig formal aussehendes Schaltbild zu zeichnen. Durch diesen Schritt sollte Fehlern beim späteren Aufnähen der Leiterbahnen vorgebeugt werden[10]. Welche Eigenschaften die Skizzen aus Modellsicht abbildeten wird in Abschnitt 7.3.3 untersucht (siehe auch Abbildung 7.6).

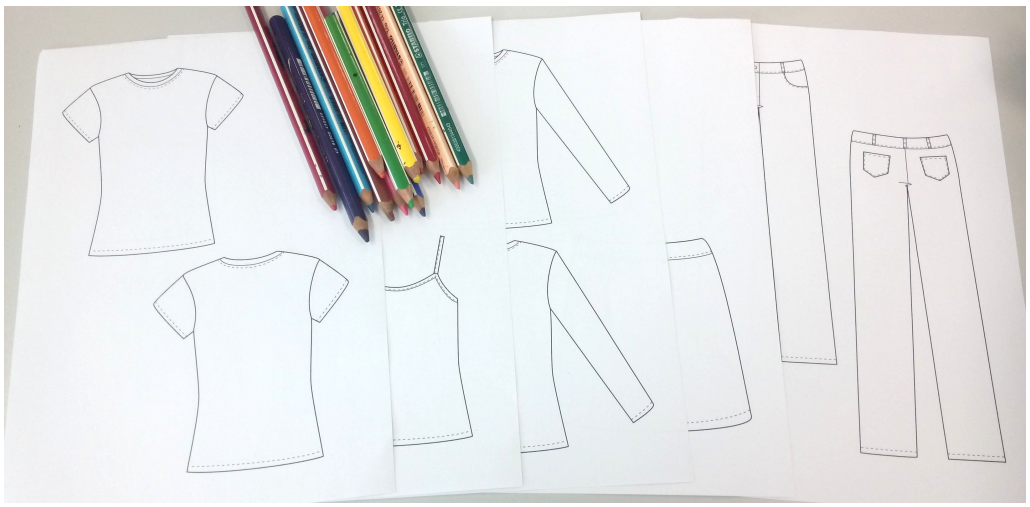


Abbildung 6.5: Vorlagen für die Skizzen und Buntstifte zum Zeichnen

Diese Methode kann am ehesten die Anforderungen *Allgemeinverständlichkeit*, *Persönlicher Kontext*, *Einfachheit* und *Praxisintegration* erfüllen. Jedoch bietet sie weniger Freiraum und *Flexibilität*. Des Weiteren wurde sie so konzipiert, dass hauptsächlich eine Perspektive auf die Schnittstelle zwischen Artefaktkorpus und Hardware, also ein statisches Element, eingenommen wird.

In der späteren Analyse werden noch weitere Skizzen ausgewertet, für die keine speziellen Vorlagen entwickelt wurden.

6.3.2 Vorspielen und Dokumentieren von Szenarien

Für einen TechKreativ-Workshop im Herbst 2009 wurde eine Methode zur Ideenfindung entwickelt, Szenarien vorzuspielen und mittels Fotos und Notizen zu dokumentieren.

Diese Technik basiert auf ähnlichen Konzepten von z.B. Binder (1999), Brandt und Grunnet (2000) oder Macaulay et al. (2006), denen das Prinzip des ‚acting out' von

[10]Beim Arbeiten mit smart textiles stellt das Anbringen des leitfähigen Garns eine Herausforderung dar, da dieses nicht isoliert ist und es somit schnell zu Kurzschlüssen kommen kann.

Szenarien zugrunde liegt – teilweise auch unter Verwendung von Mock-ups, die das Artefakt repräsentieren, für das es die Interaktion zu gestalten gilt.

Die Methode sollte dabei helfen, allgemeinverständliche, nachvollziehbare Modelle der Interaktion zu erstellen. Ferner wurde mit ihr exploriert, wie sich Projektideen abbilden lassen, indem sie durch Fotos und Notizen auf die wesentlichen ‚Schlüsselmomente' reduziert werden. Durch das Vorspielen sollten Ideen nicht nur angeregt werden, sondern konkret durchlebt werden, so dass schon bei der Ideenfindung der Interaktionsspielraum bedacht bzw. ‚erlebt' und ausagiert wird.

Um beim Vorspielen ‚etwas in der Hand zu haben', das das zukünftige Artefakt bzw. dessen Computer repräsentiert, wurden Mock-Ups aus schwarzen Pappschachteln in zylindrischer flacher Form von ca. sieben Zentimetern Durchmesser mit einem optionalen, abnehmbaren Band (z.B. zum Umhängen oder Befestigen) verteilt. Die Mock-ups waren bewusst neutral gestaltet, so dass wenig Assoziationen zu existierenden Gegenständen erkennbar werden, es aber trotzdem die Form eines LilyPads hatte, mit dem die Artefakte später umgesetzt wurden.

Abbildung 6.6: Materialien zur Dokumentation von Szenarien: Stift, Kamera, Notizzettel, Mockup mit Befestigungsschnur (im Uhrzeigersinn)

Die Teilnehmenden wurden gebeten, mit den Mock-ups Ideen für Spuk-Objekte[11] zu entwickeln und diese dabei zu fotografieren oder zu filmen. Den Workshopteilnehmenden wurden in Dreiergruppen jeweils ein Fotoapparat, Stift und Notizzettel sowie das Mock-up (siehe Abbildung 6.6) ausgehändigt, so dass jede_r im Team eine Rolle und Aufgabe übernehmen konnte (diese konnten aber auch getauscht werden). Bei der späteren Implementierung ihrer Projekte hatten sie die Fotos und Notizen vorliegen. Eine der Dokumentationen ist in Abbildung 7.2 (Seite 144) zu sehen.

Dieser Ansatz erfüllt am ehesten die Anforderungen *Allgemeinverständlichkeit*, *Persönlicher Kontext*, *Einfachheit* und *Praxisintegration*. Die Methode bietet relativ viel Freiraum und ist damit *flexibel*. Sie ist so gestaltet, dass vordergründig eine Perspektive auf die Interaktion eingenommen wird, also nicht das Artefakt als Ganzes abgebildet wird.

[11] Das Thema des Workshops lautete „Spuk an der Uni".

6.4 Amici – Modellieren mit einer Programmierumgebung

Ein wesentlicher Teil des Modellierungsprozesses stofflich-digitaler Artefakte ist deren Programmierung. Diese erst macht sie interaktiv und gibt ihnen ihre Besonderheit als Digitale Medien. Zumal ist dies neben der Hardwarearchitektur der Vorgang, der Formalisierung in Hinblick auf das erdachte Verhalten des Artefakts erfordert. Hierbei entstehen Zielkonflikte – wie in Abschnitt 3.3 beschrieben. Deshalb ist bei der Überlegung, wie Modellierungsprozesse des Selbermachens stofflich-digitaler Artefakte unterstützt werden können, auch die Wahl und die Konzeption der Programmierumgebung von Bedeutung. Im Rahmen dieser Arbeit wurde die Visuelle Programmierumgebung *Amici* hinsichtlich der genannten Anforderungen und Überlegungen weiterentwickelt.[12]

6.4.1 Ausgangslage

Die graphische Programmierumgebung *Amici* wurde ursprünglich im Rahmen des Projekts EduWear (Katterfeldt, Dittert & Schelhowe, 2009) und im Rahmen einer Dissertation (Reichel, 2008) initiiert. Auch nach Ende der Projektförderung 2008 wurde Amici kontinuierlich weiterentwickelt, u.a. auf Grundlage von protokollierten Beobachtungen aus TechKreativ-Workshops. Auf Anfrage von Nutzenden aus aller Welt wurde Amici mit ihrer Hilfe – neben den ursprünglichen Sprachen Deutsch und Englisch – ins Spanische, Portugiesische, Französische, Dänische und ins Niederländische übersetzt.

Amici basiert technisch auf der Programmierumgebung Arduino[13] und stellt somit eine Erweiterung dieses Programmiereditors um ein zusätzliches graphisches Interface dar. Daher steht Amici (wie auch Arduino) unter GNU-GPL-Lizenz[14] und wird zum kostenfreien Download angeboten.[15] Amici und Arduino sind in Java programmiert.

Ausgangslage für die unten beschriebenen Erweiterungen waren die Versionen von Amici, die bis Anfang 2010 (zuletzt unter der Versionsnummer amici0017l) veröffentlicht wurden.[16] Um die jeweiligen Zustände der Software vor und nach den mit dieser Arbeit begründeten Erweiterungen benennen zu können, wird im Folgenden zwischen ‚Amici-2009' (siehe Abbildung 6.7) und ‚Amici' unterschieden. ‚Amici' bezeichnet den Entwicklungsstand seit 2012 (siehe Abbildung 6.8).

[12] An der Entwicklung der VPL Amici wirkten zeitweise zwei Kolleginnen mit, siehe Reichel (2008) und Dittert (2015). Die Autorin dieser Arbeit war seit Ende 2007 beteiligt.

[13] `http://arduino.cc/en/main/software` (aufgerufen am 02.03.2014)

[14] siehe `http://www.gnu.de/documents/gpl.de.html` (aufgerufen am 01.04.2012)

[15] Aktuelle Versionen unter `http://dimeb.de/eduwear/tag/amici/` (aufgerufen am 02.03.2014)

[16] Bei den Neuerungen zwischen 2007 und 2010 handelte es sich vorwiegend um Fehlerbehebungen, technische Aktualisierungen und Übersetzungen des Interfaces in weitere Sprachen.

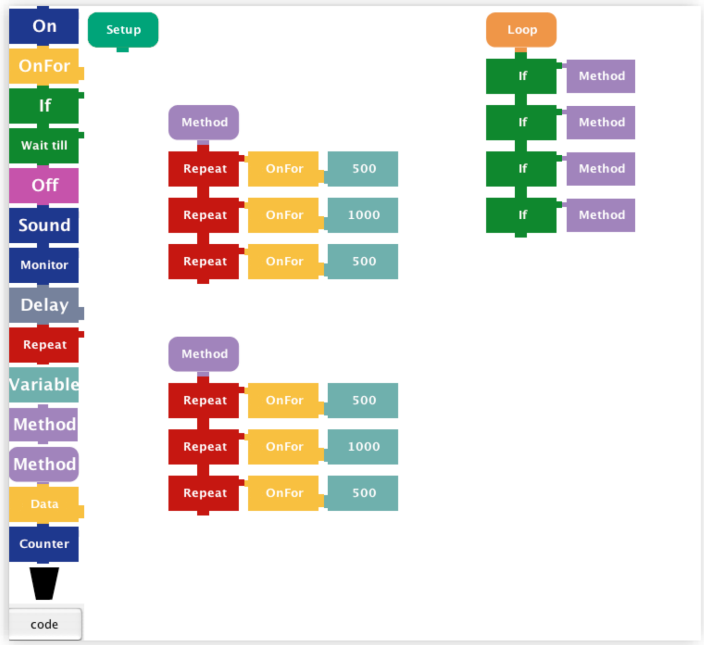


Abbildung 6.7: Visueller Programmcode in Amici-2009 (von ca. 2007 bis Version amici0017l im April 2010)

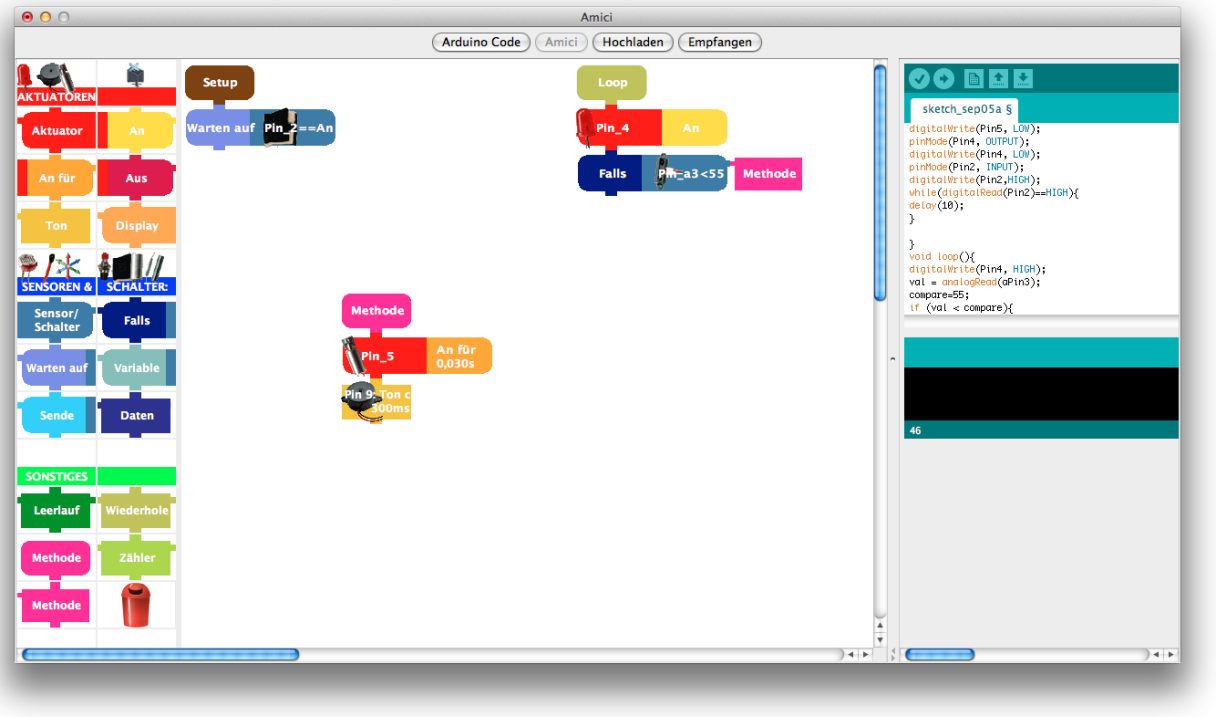


Abbildung 6.8: Amici mit Hardware-Icons auf Blöcken und sichtbarem Textcode (veröffentlicht als Version amici0022p im März 2012)

6.4.2 Überarbeitung im Rahmen dieser Arbeit

Die hier vorgestellten Weiterentwicklungen von Amici-2009 wurden von der Autorin vorgenommen, um die Software als ein weniger formal erscheinendes, allgemeinverständliches Modellierungswerkzeug zur Programmierung der Arduino-Boards zu optimieren.

Integration stofflicher Hardwarekonfiguration (2010): In Workshops beobachtete die Autorin wiederholt, dass Teilnehmende bei der Vorbereitung ihrer Projektpräsentation nicht mehr in der Lage waren, eine Verbindung zwischen dem Amici-Code, den sie programmiert hatten, und ihrem Artefakt herzustellen. Sie konnten nicht mehr benennen, was am Artefakt z.B. ‚an' gemacht wurde. Deshalb wurden Bilder von üblichen Sensoren und Aktuatoren in Amici integriert. Sie können ausgewählt werden, sobald ein Programmierblock in das Programm gezogen wird und sind dann auf dem entsprechenden Block sichtbar (siehe Abbildung 6.8). So können Betrachtende sofort erkennen, auf welchen Sensor oder Aktuator sich das Programmierkommando dieses Blocks bezieht.

Mit der Integration ikonischer Abbildungen der Sensoren und Aktuatoren wurde eine Verknüpfung zur konkreten, stofflichen Hardware geschaffen. Der ‚reale' Kontext, bzw. Informationen zur Perspektive auf die Hardwarearchitektur des Artefakts, wird so in den ‚virtuellen' Teil des Projekts – das Programm – eingebunden.[17] Auch wurden im Zuge der Überarbeitung zusätzliche textuelle Informationen (als abundante textuelle Modelle) einzelnen Blöcken hinzugefügt, wie Pinnnummern, an denen Sensoren und Aktuatoren angeschlossen sind, sowie Methodennamen, Operatoren und zusätzliche Parameter. Dies hatte zur Folge, dass aus Platzgründen die Programmbefehle, die sich auf Inputs (Sensoren und Schalter) und auf Outputs (Aktuatoren) beziehen, auf zwei Blöcke aufgeteilt werden mussten. Um einen Aktuator anzuschalten, muss nun zunächst ein Aktuatorblock gesetzt und um einen Aktionsblock (z.B. „an") ergänzt werden. Die Blöcke sind farblich codiert: der Aktuatorblock und zugehörige Aktionsblöcke in Rottönen, Sensor/Schalterblöcke und zugehörige Aktionsblöcke in Blautönen. Durch die Unterscheidung sollen die Programmierenden leichter zwischen Sensor-Input und Aktuator-Output entscheiden können (was sich in Beobachtungen mit Amici-2009 zuvor oft als schwierig erwiesen hatte). Auch im Blockmenü (links auf der Programmieroberfläche) sind Sensorblöcke und Aktuatorblöcke entsprechend gruppiert und durch Icons der Hardwarekomponenten illustriert, so dass Programmierkommandos anhand der zu programmierenden physischen Komponente identifiziert werden können.

[17] Die Integration von Icons von Hardwareelementen wurde von der Autorin schon im Modellierungstool *MoTo* konzipiert und implementiert, um Kontext zu integrieren (siehe Abschnitt 5.3.1 und Katterfeldt und Schelhowe (2008)). Auch Reichel (2008) zeigte, dass das Hinzufügen von Schlagworten zu Blöcken in Amici-2009 helfen kann, persönlichen Kontext in die Programmierung einfließen zu lassen.

Neben der *Allgemeinverständlichkeit* durch den Bezug zur Realität über ikonische Modelle wurde mit der Integration von Aktuator- und Sensorbildern auch die *Perspektivenvielfalt* durch Ergänzung der Perspektive auf die Hardware erhöht, wenn auch beide Perspektiven nicht voneinander getrennt dargestellt werden. Außerdem wird im Sinne einer *Modellschnittstelle* der Zusammenhang zwischen Hardwarekonfiguration und Programmierung abgebildet.

Sichtbarkeit von Textcode (2012): In einer weiteren Iteration wurde 2012 von der Autorin ein Fenster für den Arduino-Code integriert, so dass dieser unmittelbar neben dem Block-Code zu sehen ist und sich während des Programmierens entsprechend aufbaut. Zuvor wurde der Code erst auf einer weiteren Ebene des Interfaces sichtbar, zu der gewechselt werden konnte, und blieb somit die meiste Zeit versteckt. Außerdem konnte so der direkte Zusammenhang zwischen einem Block und dem entstehenden neuen Code nicht beobachtet werden. Jetzt wird der textuelle Code in Echtzeit automatisch erzeugt, sobald ein Block auf der visuellen Oberfläche in den Programmablauf eingereiht wird. Damit wird das abstraktere Modell, das sich hinter den Amici-Blöcken verbirgt, sichtbar und die Amici-Blöcke ‚transparenter'. Die Übergänge und Zusammenhänge zwischen Blöcken und textuellem Code werden in dieser *Modellschnittstelle* offengelegt. Den Nutzenden wird eine neue Perspektive auf den abstrakten Code (*Perspektivenvielfalt*) eröffnet, ohne dass sie aktiv in die Code-Ansicht wechseln müssen. In Abbildung 6.8 ist das aufgrund der genannten Überlegungen überarbeitete Amici (Stand Mai 2013) zu sehen.[18]

Insgesamt trägt die erweiterte Amici-Umgebung dazu bei, das Verhältnis verschieden abstrakter Modelle durch das sichtbare Code-Fenster offen zu legen. Auch schafft sie einen konkreten Bezug zur gegenständlichen Hardware, die die Programmierenden bereits – so wird angenommen – händisch zusammen gesteckt und vor sich liegen haben. Damit kann Amici an konkreten, gegenständlichen Erfahrungen anknüpfen.

Andere Visuelle Programmierumgebungen für Arduino, die in Abschnitt 5.3.4 beschrieben wurden, sind (mittlerweile) ähnlich gestaltet wie Amici. Minibloq besitzt seit 2011 ein geteiltes Fenster mit Blöcken und Programmcode. Modkit und Minibloq integrieren mittlerweile auch GUI-Elemente für die Hardwarekonfiguration.

6.5 MoLab – Dokumentieren von Modellierungsprozessen

Mit dem VirtualLab und der VPL Amici bestanden schon Tools, die beim Selbermachen stofflich-digitaler Artefakte in Workshops eingesetzt wurden. In Amici (siehe Abschnitt 5.3.5) erfolgt ein wesentlicher Modellierungsschritt durch das Erstellen des formalen Programms, das durch die Visuelle Programmiersprache relativ übersichtlich und auch

[18] Diese Version unterstützt im Vergleich zu Amici-2009 u.a. weitere Sensoren, die im Rahmen des Promotionsprojekts „TechSportiv" von Dittert (2015) ergänzt wurden.

für Anfänger_innen bald verständlich ist. Die VPL bietet dabei eine hohe Praxisintegration, da die Artefakte unweigerlich programmiert werden müssen. Für diese Arbeit wurde Amici u.a. so weiterentwickelt, dass es einen größeren Bezug zur Hardwarekonfiguration zulässt. Es bietet jedoch vorwiegend Modellierungswerkzeug an, um formale Aspekte des Artefakts abzubilden und es zu programmieren.

Das VirtualLab (siehe Abschnitt 5.1.2) erlaubt mit allgemeinverständlichen Mitteln – hauptsächlich Fotos und Text – ein Projektergebnis zu beschreiben. Es erlaubt auch das Hochladen des Programmcodes und das Auswählen der verwendeten Hardwareelemente in der Eintragungsmaske, so dass verschiedene Aspekte des Artefakts abgebildet werden können. Die Tabelle in Abbildung 6.1 zeigt eine Einschätzung, inwiefern die beiden Tools die aufgestellten Anforderungen erfüllen können. Jedoch zeigt die Tabelle auch, dass Amici und das VirtualLab verhältnismäßig unflexibel sind und nicht den Modellierungsprozess darstellen. Auch lässt sich mit ihnen – aus konzeptioneller Sicht – das Artefakt, das gerade konstruiert wird, nur aus wenigen Perspektiven abbilden. Um die aufgestellten Anforderungen zu überprüfen, wurde deshalb die Webplattform MoLab als ein weiteres Tool konzipiert und implementiert, das sich an How-to- und Storyboard-Prinzipien anlehnt. Es ist flexibler als die anderen Tools, indem es mehr Modellierungsfunktionen bereit stellt, und es ermöglicht das Abbilden von vielfältigen, verketteten, nach Perspektiven getrennten Modellen. Dieses Tool wird in Anlehnung an das VirtualLab *MoLab* (für Modell-Lab) genannt.

	Allgemeinverständlichkeit	**Persönlicher Kontext**	**Flexibilität**	**Praxisintegration**	**Einfachheit (Erstellen von Modellen)**	**Perspektivenvielfalt**	**Perspektivtrennung**	**Modellnetze/ -folgen**	**Modellschnittstellen/ -übergänge**
Virtual Lab	Bilder, Text	möglich z.B. durch Fotos	niedrig: Dateiformate und Bauteile beschränkt	wenig: Modelle erstellen ist zusätzlicher Aufwand	gering: erfordert Login und Transfer von Bild und Programm	Projektdoku mit Sensoren/Aktuatoren, Programm-Datei, Bild, Text	bedingt: Programm und Aktuatoren/Sensoren werden separat eingetragen	nicht möglich, nur fertiges Projekt/ Artefakt	Zusammenhang erkennbar, da alle Modelle fertiges Artefakt abbilden
Amici (2012)	weniger umgangsspr.; visueller Code und Hardware-Icons, aber übersichtlicher als textueller Code	eher wenig (visueller Bezug zur Realwelt über Hardware-Icons)	niedrig: keine anderen Modelle außer vorgegebenen Elementen möglich	hoch Programmierung muss sowieso gemacht werden	Programmierung relativ einfach und unkompliziert; aber keine anderen Modelle möglich	Vis. Programm mit Hardware-Informationen und Textcode	Fokus auf Programmierung; Informationen zu Hardwarekonfig. im vis. Programmcode	nicht möglich	automatisch zwischen Hardware & Programmierung, vis. Code & Textcode
MoLab	Bilder, Text, Storyboarding	hoch (Einbezug von Fotos, freie Anordnung)	hoch: versch. Dateiformate, versch. Anordnungsmöglichkeiten	eher gering: Modelle erstellen ist zusätzlicher Aufwand	gering: erfordert Login und Transfer von Programm sowie Arbeitsunterbrechungen	Programm (Datei), Hardware, Korpusgestalt, Interaktion	ja (in Version 1.0)	ja	möglich durch eigene Gruppierung von Einträgen

Tabelle 6.1: Einschätzung der Tools hinsichtlich der Erfüllung der Anforderungen anhand vorhandener Funktionen

Wie in Abschnitt 2.5.1 beschrieben wurde, dokumentieren Maker_innen ihre Projekte oft als How-tos, wie sie z.B. auf Plattformen wie Instructables (siehe Kapitel 5) zu finden sind. Diese How-tos ähneln strukturell Storyboards, indem ein Bild zusammen mit Text eine Schlüsselszene als einen Projektschritt beschreibt. Im Makerumfeld können How-tos als etablierte Praxis für das Dokumentieren und das Austauschen von Projekten und zum Lernen voneinander angesehen werden. Anders als Storyboards, die wie Comics normalerweise von links nach rechts notiert werden, werden How-tos von oben nach unten angeordnet (vermutlich aus Gründen der Scrollbarkeit im Internetbrowser). How-tos dokumentieren bisher nur fertige Projekte und dienen damit als Anleitung, die im Nachhinein den Modellierungsprozess abbildet. Andererseits folgen How-tos und Storyboards einer linearen Struktur, die nicht unbedingt dem kreativen Prozess entspricht, in dem Maker_innen Artefakte erstellen (siehe Abschnitt 5.1.3).

Anhand von MoLab soll auch untersucht werden, ob sich ein Dokumentieren mit vielen kleinen Modellen (statt abgeschlossener Schritte wie im How-to) zum Abbilden von Artefakt und Modellierungsprozess anwenden lässt. In einem typischen How-to enthält meist jeder Schritt eine in sich abgeschlossene Tätigkeit, z.B. Zusammenbau der Hardware, Bau des Artefakts, Programmierung usw. Da dieses Vorgehen für einen Konstruktionsprozess, der noch nicht abgeschlossen ist und sich noch entwickelt, eher unwahrscheinlich erscheint, bietet MoLab die Möglichkeit, viele einzelne Modelle mit verschiedenen Perspektiven zu erstellen und diese entsprechend zu gruppieren. So können mit MoLab verschiedenartige Modelle eines Projekts dokumentiert werden: Modelle, die verschiedene Aspekte eines Artefakts (bzw. seine Teilsysteme) abbilden, können gruppiert werden, so dass diese Gruppierung ein (umfassendes) Modell des Artefakts abgibt (ähnlich der Projektdokumentationen im VirtualLab). Alle Modelle, die während der Dauer eines Projekts dokumentiert wurden, bilden zusammengenommen wiederum auch ein Modell des Modellierungsprozesses.

MoLab stellt ein Spektrum von Funktionen zur Verfügung, um eine breite Auswahl an Möglichkeiten zum Abbilden von Modellen in der Praxis explorieren zu können. Damit entspricht MoLab bewusst nicht der Anforderung nach Einfachheit, ist aber flexibler als die anderen Tools.

MoLab wurde anhand der im Theorieteil dieser Arbeit gewonnenen Erkenntnisse konzipiert und ist geprägt durch Erfahrungen der Autorin mit TechKreativ-Workshops (siehe Abschnitt 2.4.2). Für die Entwicklung von *MoLab* wurde auf aktive Nutzerbeteiligung in der Konzeptionsphase verzichtet. Die Gründe liegen in der theoretischen Ausrichtung dieser Arbeit und dem damit verbundenen Zweck des Tools, die Anforderungen zu implementieren und in der Anwendung zu überprüfen.

Im Folgenden wird Version 1.0 von MoLab vorgestellt. Diese wurde während und nach dem ersten Workshop iterativ – entsprechend des design-based-research-Ansatzes (siehe Abschnitt 6.2.2) – weiterentwickelt. Die sich daraus ergebenden Ände-

rungen sind in Abschnitt 7.4.6 beschrieben. Hier wird zunächst der Aufbau von MoLab vorgestellt.

6.5.1 Funktionen, Gestaltung und technische Entwicklung

Mit *MoLab* wurde ein Modellierungstool erstellt, das sich einer Storyboard-Metapher bedient. MoLab ist wie das VirtualLab eine Webanwendung, die auch Austausch unter den Nutzenden ermöglicht.

Registrierung, Login/Authentifizierung: Um das System nutzen zu können, muss ein Account erstellt werden. Mittels Passwort und Username kann sich die bzw. der Nutzende einloggen. Sämtliche Inhalte des Systems können nur nach Authentifizierung genutzt werden, um Missbrauch zu vermeiden.

Projekt-Übersicht: Die Nutzenden erhalten zuerst eine Übersicht über alle aktuellen Projekte, die neben den jeweiligen Projekttiteln auch die zugehörigen Bilder zeigt (siehe Abbildung 6.9). Registrierte Nutzer_innen können alle Projekte ansehen und öffnen (für einen Überblick der Navigationsstruktur siehe Abbildung 6.10).

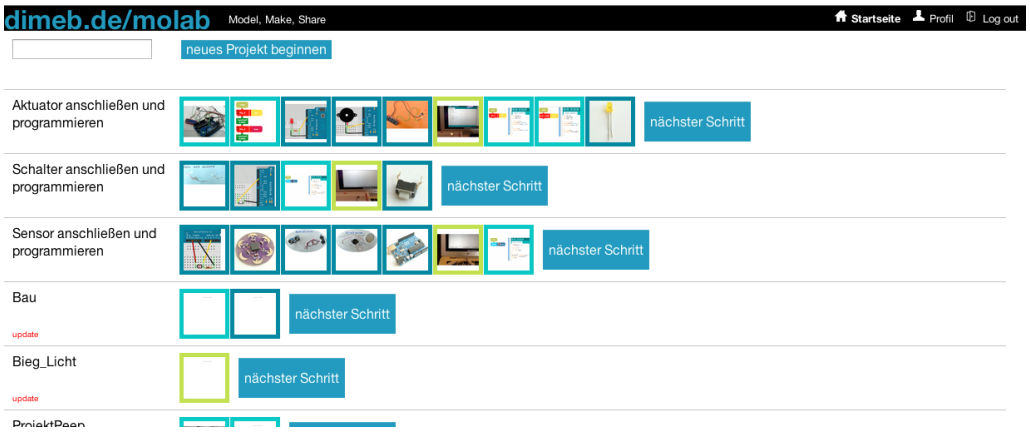


Abbildung 6.9: Startseite mit Projekten in MoLab

Das Modelboard: Das Interface weist in Version 1.0 vier Spalten auf, eingeteilt nach Nutzer-Interaktion, Programm, Hardware und Design (siehe Abbildung 6.11).[19] In den Spalten befinden sich die jeweils zugeordneten MoLab-Einträge. Diese können vertikal, innerhalb ihrer Spalte (dem Modelstrang) verschoben werden. Zwischen den MoLab-Einträgen können Lücken gelassen werden, um z.B. eine horizontale Verbindung zu Elementen der anderen Modellstränge visuell herzustellen (siehe Abbildung 6.12).

[19] Im Gegensatz zu den Perspektiven in Abschnitt 6.1.3 wurden umgangssprachliche kurze Begriffe gewählt. Design soll für die Gestaltung des Artefaktkorpus stehen, Nutzer-Interaktion für das interaktive Verhalten.

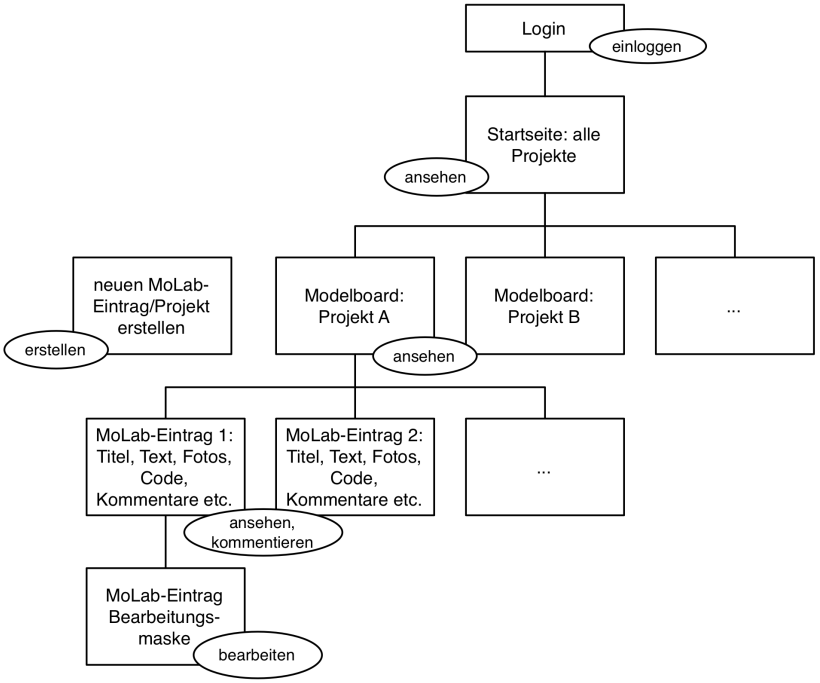


Abbildung 6.10: Sitemap: Seitenfolge in MoLab

Ein MoLab-Eintrags-Element zeigt sein erstes Bild und alternativ den Text des Eintrags. Sind mehrere Bilder vorhanden, kann durch alle Bilder gescrollt werden. Annotationen des Bildes und Beschreibungstext des MoLab-Eintrags können ein- und ausgeblendet werden. Weitere Icons zeigen an, ob ein Video hochgeladen wurde, und ob es sich bei dem Eintrag um ein unfertiges work-in-progress oder einen fertigen Schritt einer Anleitung handelt. Außerdem wird die Anzahl neuer Kommentare angezeigt. Unten befinden sich Buttons zum Speichern der per drag-and-drop geänderten Reihenfolge und Anordnung der MoLab-Einträge. Außerdem können die MoLab-Einträge mittels einer Zoom-Funktion in ihrer Höhe angepasst werden, so dass bei vielen Modellelementen eine bessere Übersichtlichkeit hergestellt werden kann. Durch Klicken auf den Titel gelangt die bzw. der Nutzende in die Ansicht des jeweiligen MoLab-Eintrags.

Erstellen und Bearbeiten von MoLab-Einträgen: In der Bearbeitungsmaske eines MoLab-Eintrags können alle oben beschriebenen Felder von dessen Urheber_in bearbeitet werden. Maximal fünf Bilder können eingebunden werden, um das Erstellen mehrerer MoLab-Einträge anzuregen. Es können nicht nur Bilder vom lokalen System hochgeladen werden. Eine Bibliothek stellt Bilder von häufig genutzten Hardwarebauteilen zur Verfügung. Auch können Bilder durch URLs eingebunden werden. Sofern der Computer, auf dem MoLab benutzt wird, über eine Webcam verfügt, kann diese direkt aus MoLab angesteuert werden. So können z.B. das Artefakt, seine Umgebung oder ein

Abbildung 6.11: Beispiel eines *Modelboards* in MoLab 1.0 mit Einträgen sortiert nach Artefaktaspekten (enthalten Screenshots von Amici und Fritzing)

.

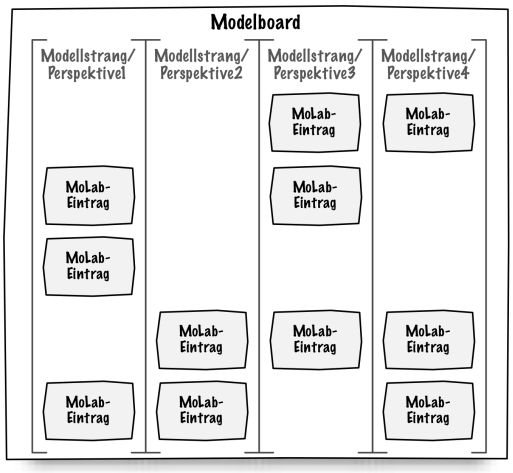


Abbildung 6.12: Struktur und Elemente des *Modelboards*

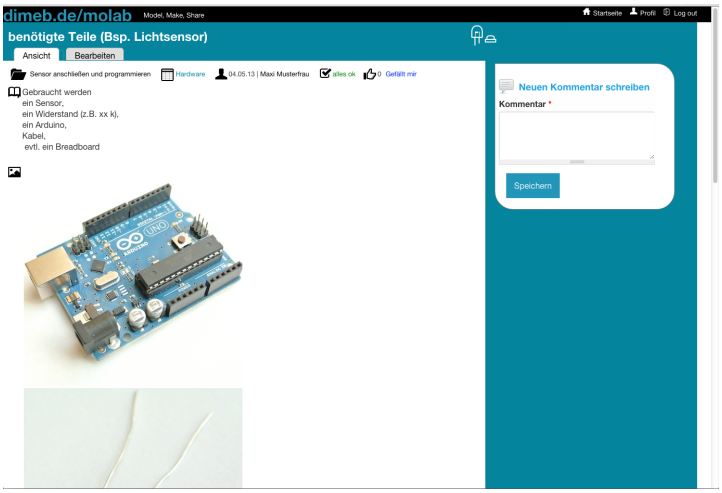


Abbildung 6.13: MoLab-Eintrag mit Text und Bildern

extern erstelltes Modell unkompliziert aufgenommen und in MoLab-Einträge eingebunden werden.

Auf bereits hochgeladene und gespeicherte Bilder können Annotationen in Form von Tags gesetzt werden. Anhänge anderen Dateiformats, z.B. Dateien mit Programmcode, können hochgeladen werden. Außerdem kann pro MoLab-Eintrag eine Videodatei (als dynamisches Modell) hochgeladen werden, die direkt in der Ansicht des MoLab-Eintrags abgespielt werden kann. Über einen „Brauche Hilfe"-Button kann ein MoLab-Eintrag entsprechend markiert werden.

Ansicht eines MoLab-Eintrags: In der Ansicht eines MoLab-Eintrags werden (sofern vorhanden) alle zugehörigen Bilder, Videos und Dateianhänge, der Name des zugehörigen Projekts, die Art der Perspektive (Hardware, Programm, usw.), die Beschreibung, die Autorennamen und das Erstellungsdatum angezeigt. Auf den Bildern werden – sofern hinzugefügt – die Annotationen mit Nummerierung eingeblendet. Rechts neben dem Eintrag erscheinen Kommentare (siehe Abbildung 6.13).

Technische Entwicklung: MoLab basiert auf dem OpenSource Content Management System *Drupal* (Version 7)[20] und benutzt diverse Module und Plugins, die teilweise für MoLab umprogrammiert wurden. Dadurch liegen der Plattform die Webtechnologien HTML5, PHP, MySQL, JavaScript und jQuery zugrunde. Der volle Funktionsumfang wurde zum Zeitpunkt der Erprobung nur von Browsern vom Typ Google Chrome[21] unterstützt.[22]

6.5.2 Konzeptionelle Annahmen

Das Modellierungstool MoLab wurde anhand der Anforderungen, die in Abschnitt 6.1.4 formuliert wurden, konzipiert. Damit gehen Annahmen und Überlegungen einher, die hier dargestellt werden.

Die Übersichtsseite eines Projekts in MoLab wird – in Anlehnung an Storyboards – *Modelboard* genannt. Auf dem Modelboard werden alle MoLab-Einträge eines Projekts aufgeführt (siehe Abbildung 6.11). Angeordnet sind sie (in Version 1.0) als *Modellstränge* in parallelen Spalten, die wiederum je eine Kategorie darstellen. MoLab-Einträge können den Kategorien Interaktion, (Produkt-)Design, Hardware und Programm zugeordnet werden. In der entsprechenden Spalte erscheinen sie auf dem Modelboard.

[20] `http://www.drupal.org` (aufgerufen am 01.06.2012)

[21] `https://www.google.de/intl/de/chrome/browser/` (aufgerufen am 19.02.2014)

[22] MoLab wurde auch für die Verwendung mit dem mobilen Apple-Betriebssystem iOS6 optimiert. Damit können auch die Kameras von Smartphones und Tablets benutzt werden, um direkt im Bearbeitungsmodus eines MoLab-Eintrags Bilder oder Videos zu erstellen und einzubinden. Aus Usabilitysicht ist dies von Vorteil beim Erstellen von Fotos. Allerdings können hierbei Dateien (z.B. Programmcode), die auf dem Smartphone oder Tablet nicht erstellt werden können, nur umständlich hochgeladen werden. Mobile Geräte und drahtlose Internetverbindungen waren aber nicht ausreichend verfügbar, so dass MoLab in Workshops nicht auf mobilen Geräten erprobt werden konnte.

Dieser Implementierung liegen die Anforderungen *Perspektivenvielfalt* und *Perspektivtrennung* zugrunde.

Mehrere MoLab-Einträge, die untereinander in einer Spalte erscheinen, bilden auch eine Modellkette, also eine Folge von Modellen, die in einem Zusammenhang zueinander stehen. Um zusammengehörige MoLab-Einträge zu kennzeichnen, können sie entsprechend gruppiert werden. Auch können durch entsprechende horizontale Anordnungen Bezüge zu Modellen aus anderen Spalten hergestellt werden (siehe Abbildung 6.12). Damit werden die Anforderungen der *Modellnetze* und *Modellschnittstellen* berücksichtigt. Mit dem Modelboard wird eine Möglichkeit angeboten, viele kleine Modelle entsprechend anzuordnen und zu gruppieren. Mit der Implementierung in MoLab soll nun überprüft werden, ob eine Einteilung einzelner Modelle in Perspektiven bzw. Kategorien, als auch die Möglichkeit diese zu gruppieren, für Modellierende Sinn ergeben und nützlich sind.

MoLab bietet verschiedene Möglichkeiten, Bilder in Einträge einzubinden. Neben dem Hochladen von Bildern können in einem MoLab-Eintrag direkt über die eingebaute (oder angeschlossene) Kamera des Rechners Fotos erstellt und unmittelbar integriert werden. Mit diesen Funktionen soll das Erstellen von Bildmodellen erleichtert und angeregt werden. Damit werden die Anforderungen *Flexibilität*, *Persönlicher Kontext*, *Einfachheit* und *Praxisintegration* berücksichtigt.

Es werden verschiedene Möglichkeiten angeboten, textuelle Modelle zu erstellen. Neben dem Titel- und Beschreibungsfeld können auch Fotos mit Stichworten annotiert und Kommentare von Nutzenden der Plattform geschrieben werden. Die Felder sind optional. Der Titel wird automatisch aus Autorenname und Datum generiert, wenn dort kein Text eingetragen wird. In den Textfeldern sollen Modellperspektiven, die sich über die anderen Medien schwierig darstellen lassen, dokumentiert werden können.

Auf die hier angeführten Annahmen wird in der Evaluation in Abschnitt 7.4.9 Bezug genommen, wo anhand einer Fallstudie untersucht wird, wie jugendliche Workshopteilnehmer_innen mit MoLab modellieren.

6.6 Zusammenfassung und Schlussfolgerungen

In diesem Kapitel wurden zunächst Rückschlüsse aus der theoretischen Betrachtung von Modellierung in Informatik- und Designkontexten gezogen und diese auf den Kontext ‚Making' von stofflich-digitalen Artefakten übertragen. Auch dort kann von Modellierung gesprochen werden. Making wurde als eher designartiger, kreativer, iterierender Modellierungsprozess eingeordnet. Insbesondere beim Selbermachen stofflich-digitaler Artefakte entsteht eine Vielfalt von Modellen, die sich immer wieder ändern, so dass innerhalb eines Projekts rückblickend nicht-lineare Modellketten oder -netze entstehen. Gleichzeitig dienen Modelle als Reflexionsobjekte, anhand derer Ideen entstehen und Projekte entwickelt werden. Modelle des zu modellierenden

oder modellierten Artefakts spielen, z.B. in Form der How-tos, auch eine wichtige Rolle in Zusammenhang des Tauschens und Lernens voneinander. Das setzt voraus, dass Modelldokumentationen ein nachvollziehbares Bild des Projekts abgeben. Dies betrifft das Darlegen von Zusammenhängen zwischen formalen Modellen und unformaler Erscheinung als auch das Mitteilen pragmatischer Merkmale wie Intentionen. Da es sich bei Modellen um reduzierte Abbilder handelt, die jeweils nur einen Ausschnitt auf ihr Original zeigen, wurden fünf geeignete Perspektiven für Modelle stofflich-digitaler Artefakte abgeleitet. Diese können nützlich sein, um die Komplexität des Systems für die Modellierenden handhabbar zu machen. Die Perspektiven beinhalten statische formalbeschreibbare Modellperspektiven auf die Hardwarekonfiguration, dynamische formale auf das Programm, aber auch eher unformale Modellperspektiven auf das interaktive Verhalten und die statische Gestalt des Artefaktkorpus. Darüber hinaus sind diese vier Perspektiven mit pragmatischen Eigenschaften (z.B. Kontext und Intentionen) verwoben als eine fünfte, übergeordnete Perspektive.

Ausgehend von den theoretischen Betrachtungen wurden Anforderungen für Modellierungstools aufgestellt. Diese sind Allgemeinverständlichkeit, persönlicher Kontext, Flexibilität, Praxisintegration, Einfachheit, Perspektivenvielfalt, Perspektivtrennung, Modellnetze und Modellschnittstellen. Da diese Anforderungen zum Teil widersprüchlich sind oder sich zumindest nicht ohne Kompromisse in ein einziges Tool integrieren lassen, wurden drei Tools für die Untersuchungen dieser Arbeit genutzt. Dies sind die existierende Webplattform VirtualLab, die existierende Programmierumgebung Amici in einer überarbeiteten Version sowie die Webplattform MoLab, die neu für diese Arbeit entwickelt wurde. Auch wurden zwei ‚analoge' Methoden vorgestellt: das Skizzieren von Stromkreisen auf Vorlagen und das Vorspielen und Fotografieren von Szenarien. Sie befassen sich vorwiegend mit den unformalen Eigenschaften der Modellierung eines Artefakts, d.h. der Gestalt seines Körpers, dem angedachten interaktiven Verhalten, sowie der Schnittstelle zwischen Hardware und Korpus, die in den vorhandenen Tools weniger Berücksichtigung finden.

Mit Amici wurde eine Visuelle Programmierumgebung erweitert, so dass die damit erstellten Programme einfacher zu lesen sind und durch Einbeziehung von Abbildungen der Hardwareelemente kontextbezogener werden. Auch werden die Anforderungen Allgemeinverständlichkeit, Perspektivenvielfalt und Modellschnittstellen dadurch stärker berücksichtigt.

Die Konzeption von MoLab ist insbesondere auf die Anforderungen Flexibilität, Modellnetze und Perspektivtrennung ausgerichtet, die bei den anderen Tools nur ungenügend ausgebildet sind. Dazu lehnt sich MoLab grob an How-tos an, allerdings unterstützt es mehr das Erstellen vieler kleiner, sich ergänzender Modelle. Es bietet eine Bandbreite von verschiedenen Möglichkeiten, graphische, textuelle und mit externen Tools erstellte Modelle einzubinden. MoLab ist ein Modellierungstool, das –

je nach Betrachtungsweise – neben vielen einzelnen Modellen auch Modelle, die das vollständige Artefakt zeigen, und Modelle des Modellierungsprozesses darstellt.

Insgesamt wurde auf theoretischer Grundlage eine Bandbreite von Tools und Methoden entwickelt. Diese setzen verschiedene Schwerpunkte hinsichtlich der Anforderungen und konzentrieren sich auf unterschiedliche Phasen und Techniken des Modellierens. Inwiefern die implementierten Anforderungen der Praxis standhalten, wird im folgenden Kapitel evaluiert.

Kapitel 7

Evaluation: Modellanalyse und Erprobung der Tools

Um die bisherigen theoretischen Konzepte in die Praxis zu übertragen, wurden Tools und Methoden zur Modellierung stofflicher-digitaler Artefakte entwickelt, die in verschiedenem Ausmaß auf den Anforderungen Allgemeinverständlichkeit, Persönlicher Kontext, Flexibilität, Praxisintegration, Einfachheit, Perspektivenvielfalt, Perspektivtrennung, Modellnetze und Modellschnittstellen beruhen. Auch wurden Perspektiven identifiziert, aus denen Modelle beim Making erstellt werden können. In diesem Kapitel werden die theoretischen Annahmen und Anforderungen überprüft. Dazu werden Modelle, die Amateur_innen u.a. mit den Methoden und Tools aus Kapitel 6 erstellt haben, inhaltlich analysiert. Außerdem wird anhand einer Fallstudie evaluiert, wie MoLab und Amici von angehenden Makern genutzt werden. Aus den Ergebnissen werden Empfehlungen für die Gestaltung von Modellierungstools abgeleitet und exemplarisch implementiert.

7.1 Zu untersuchende Fragestellungen

Geleitet werden die Untersuchungen in diesem Kapitel von den Fragen:

- Was kennzeichnet die in TechKreativ-Workshops dokumentierten Modelle, welche Perspektiven enthalten diese und wie sind die Modelle nach AMT einzuordnen?
- Wie werden formale Eigenschaften abgebildet?
- Wie bewähren sich die von MoLab implementierten Anforderungen in der Praxis?

- Welche Elemente oder Funktionen sollte ein Modellierungstool mindestens enthalten, so dass wesentliche Aspekte des Artefakts nachvollziehbar abgebildet werden können?

Auch werden die konzeptionellen Annahmen, die MoLab betreffen (siehe Abschnitt 6.5), überprüft. Insgesamt sollen die Untersuchungen vor allem zur Beantwortung der Forschungsfrage beitragen, wie ein umfassendes und nachvollziehbares Abbilden während des Konstruktionsprozesses unterstützt werden kann und wie entsprechende Tools gestaltet sein sollten (siehe Kapitel 1).

7.1.1 Modellbegriff

Anhand des sehr allgemeinen Modellbegriffs, auf den sich diese Arbeit bezieht, können unzählige Artefakte und Konzepte, die uns umgeben, als Modelle bezeichnet werden. Deshalb sind für die nachfolgenden Untersuchungen verschiedene Ebenen des Modellierens zu unterscheiden. Zum einen wird beim Making ein Artefakt konstruiert, das selbst als ein Modell die Projektidee verkörpert und Gegenstand des Modellierungsprozesses ist. Zum anderen entstehen weitere Modelle, die Teilaspekte des Artefakts abbilden. So muss ein Programm erstellt werden; möglicherweise entstehen Skizzen zur Planung oder Fotos zur anschießenden Dokumentation. Als Modell soll im Folgenden nicht das Artefakt, das konstruiert wird, gelten, sondern externe Repräsentationen, die das Artefakt in Modellen abbilden. Das können der Programmcode (aber nicht das Programm in der Ausführung), eine Skizze, ein Foto vom entstehenden Artefakt, ein Video oder ein Text sein. Auch ein Foto einer Skizze oder ein Screenshot des Programmcodes sollen hier zu den Modellen gezählt werden, so lange diese weitestgehend isohyl und isomorph, und damit nahezu Kopien des Modells sind. Somit soll ein Foto, das hauptsächlich eine Skizze zeigt, als Skizze gelten.

Der aktuelle Zustand des Artefakts, das gerade konstruiert wird, kann als konkretes Modell begriffen werden, an dem die Projektidee reflektiert wird. Gleichzeitig ist es – neben der Projektidee – auch ein Original für Modelle. In den nachfolgenden Untersuchungen werden nur Modelle berücksichtigt, die Aspekte dieses Artefakts oder der Projektidee abbilden und die zeitlich und lokal losgelöst vom Original – z.B. über ein Modellierungstool – anderen zugänglich gemacht werden können.

7.2 Qualitative Inhaltsanalyse als Auswertungsmethode

Um die Modelle zu analysieren und das Vorgehen beim Modellieren mit MoLab auszuwerten, wurden Analyseverfahren in Anlehnung an die qualitative Inhaltsanalyse des Psychologen, Soziologen und Pädagogen Philipp Mayring (2010) gewählt. Den Grundgedanken der qualitativen Inhaltsanalyse fasst Mayring zusammen:

> „Qualitative Inhaltsanalyse will Texte systematisch analysieren, indem sie das Material schrittweise mit theoriegeleitet am Material entwickelten Kategoriensystemen bearbeitet.“ (Mayring, 2010, S. 114)

Die Inhaltsanalyse folgt einer festgelegten Struktur und zuvor definierten Regeln, so dass das Vorgehen nachvollziehbar wird und die Interpretation des Materials nicht willkürlich erscheint. Im Vergleich zu quantitativ vorgehenden Inhaltsanalyseverfahren berücksichtigt die qualitative Inhaltsanalyse z.B. auch Kontexte des Materials oder nicht ganz offenkundige Sinnstrukturen, um die Vorteile einer Systematik mit der eines qualitativen Vorgehens zu verbinden. Die qualitative Inhaltsanalyse nimmt eine Zwischenposition zwischen quantitativer und qualitativer Methode ein: Ergebnisse werden oft quantitativ aufbereitet, z.B. anhand von Häufigkeiten, während die Zuordnung der Kategorien zu Textbestandteilen durch Interpretation, also qualitative Auswertung erfolgt. Im Vorfeld wird ein Ablaufmodell festgelegt, das die Nachvollziehbarkeit gewährleistet. Es dient den Auswertenden als Leitlinie und Rahmung, um ihr Vorgehen zu begründen und zu rechtfertigen (Mayring, 2010).

Ein Aspekt der (qualitativen) Inhaltsanalyse, der sie für diese Arbeit anwendbar macht, ist ihr theoriegeleitetes Vorgehen mit Orientierung an einer Fragestellung und Kategorienbildung. In Abschnitt 6.1.3 wurden Kategorien für von Maker_innen erstellte Modelle bereits erarbeitet in Form der Modellperspektiven. Außerdem wurden in Kapitel 3 Modellkategorien und -eigenschaften der AMT vorgestellt. Diese theoretischen Konzepte sollen als Ausgangslage für die Erstellung eines Kategoriensystems für die Auswertungen dienen.

7.2.1 Skalierende strukturierende Inhaltsanalyse

Für die inhaltliche Analyse von Modellen wurde die Unterform der „skalierenden Strukturierung“ (Mayring, 2010, S. 101) gewählt, da diese sich eignet, Perspektiven und Eigenschaften von Modellattributen stofflich-digitaler Artefakte auch quantitativ zu erfassen. Diese Methode geht deduktiv vor nach einem vor der Analyse erstellten Kategoriensystem. Aus der zugrunde liegenden Theorie werden Dimensionen abgeleitet und in Kategorien ausdifferenziert. Ziel ist es, das vorhandene Material auf einer Skala einzuordnen und abzubilden. Dazu werden zu den Einschätzungsdimensionen „Variablen mit Ausprägungen in mindestens ordinalskalierter Form“ (Mayring, 2010, S. 101) definiert, die eine Skala begründen, auf der Fundstellen (mittels qualitativer Analyse) eingeordnet werden und letztendlich quantitativ z.B. nach Häufigkeit analysiert werden können.

Abbildung 7.1 zeigt das in Anlehnung an Mayring (2010, S. 93-102) entwickelte Ablaufmodell für die Modellanalyse, dessen linker Zweig das Vorgehen bei der skalierenden Strukturierung vorgab.

7.2.2 Inhaltlich-strukturierende Auswertungsmethode

Für die Untersuchung der Praxistauglichkeit von MoLab galt es mehr über die Modellierungsprozesse und -materialien und das Handeln Selbermachender herauszufinden. Deshalb wurde für die Auswertung der Modellierpraxis mit MoLab die „inhaltliche Strukturierung“ (Mayring, 2010, S. 98) als eine weitere Methode der Inhaltsanalyse gewählt, deren Ziel es ist „bestimmte Themen, Inhalte, Aspekte aus dem Material herauszufiltern und zusammenzufassen“ (Mayring, 2010, S. 98). Ausgangslage bilden auch hier theoriegeleitete Kategorien und Unterkategorien. Im Gegensatz zur skalierenden Strukturierung werden die Kodierungskategorien und Dimensionen jedoch weniger detailliert und nicht als Skala definiert. Nach dem Materialdurchlauf werden folglich nicht Häufigkeiten analysiert, sondern das Material wird paraphrasiert und nach Kategorien inhaltlich zusammengefasst (Mayring, 2010).

Das Vorgehen bei der Auswertung orientierte sich ebenfalls am Ablaufmodell in Abbildung 7.1, folgte im vorletzten Schritt aber dem rechten Zweig der inhaltlichen Strukturierung.

7.2.3 Methodische Einschränkungen

Zu der vorgestellten Methode ist zu bemerken, dass in dieser Arbeit die Inhaltsanalyse nach Mayring nicht nur auf Texte, sondern auch auf Bilder und Abbildungen von (visuellem) Programmcode angewandt wird, während Mayring (2010) sich auf die Auswertung von Texten, also fixierte Kommunikation, die in Sprachform vorliegt, bezieht. Generell bemerkt Mayring jedoch, dass jegliches kommunikationsabbildendes symbolisches Material, wie z.B. auch Bilder oder Musiknotation, Gegenstand einer Inhaltsanalyse sein kann (vgl. Mayring, 2010, S. 12). Mayrings Methode lässt sich also auch auf die im Kontext dieser Arbeit vorliegenden Materialien anwenden.[1]

Herausforderungen an Evaluation im HCI-Kontext

So wie die zunehmende Einbettung informatischer Systeme in soziale Kontexte und Lebenswelten die Herangehensweise an Modellierung geändert hat (siehe Kapitel 4), so haben sich auch die Anforderungen an die Evaluation von HCI-Systemen gewandelt. Ben Shneiderman (2008) beschreibt dies als „Science 2.0“ (Shneiderman, 2008, S. 1349) und spricht sich gegen die Anwendung traditioneller wissenschaftlicher Versuchsaufbauten aus den Natur- oder Ingenieurwissenschaften, wie Laborumgebungen, aus und fordert für die Informatik eine stärkere Hinwendung zu den Herangehensweisen der Sozialwissenschaften. Das Verortetsein von HCI-Systemen in einem interdisziplinären

[1] Eine Betrachtung von Analysemethoden visueller Kommunikation, z.B. von Müller (2003), ergab keine passende Beschreibungs- oder Analysemethode. Diese beziehen sich eher auf Bilder aus anderen Kontexten, z.B. politisch motivierte Pressebilder oder künstlerisch motivierte Bilder mit einem weiten Interpretationsspielraum. Damit erschienen solche Bildanalysemethoden für den vorliegenden Fall ungeeignet.

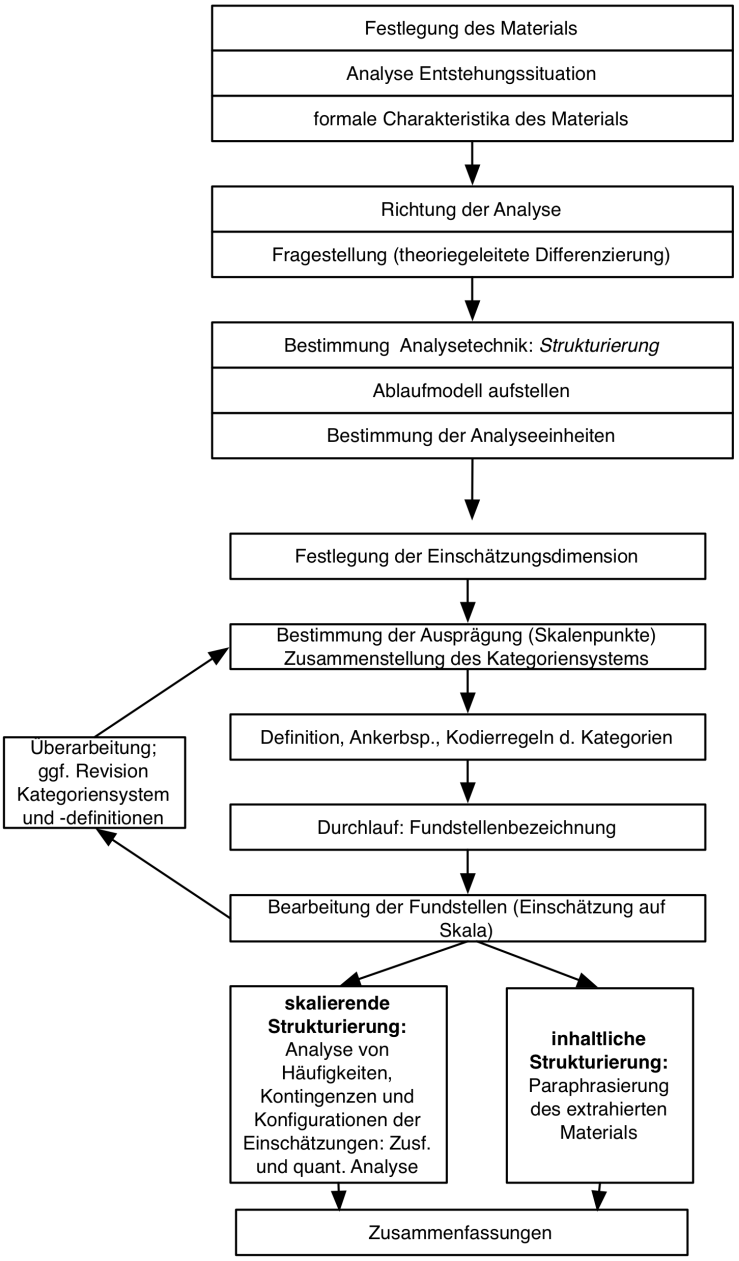


Abbildung 7.1: Ablauf der Inhaltsanalysen nach skalierender (linker Zweig) und nach inhaltlicher Strukturierung (rechter Zweig) (in Anlehnung an Abbildungen in Mayring (2010), S. 93-102)

Gebiet zwischen Engineering, Sozialwissenschaften und Design stellt somit auch eine Herausforderung an die Wahl geeigneter Erhebungs- und Auswertungsmethoden dar.

Die oben beschriebenen Auswertungsmethoden entstammen den Sozialwissenschaften. Auch in dieser Arbeit geht es darum, Prozesse menschlicher Interaktion zu verstehen – in ihrer Rolle als Nutzende von Tools, aber auch als Entwickelnde von Artefakten. Deshalb sind solche Methoden für die anstehenden Untersuchungen angemessen.

Im ersten Teil der Evaluation werden Modelle von bzw. für stofflich-digitale Artefakte untersucht. Dafür wurde ein qualitativ-quantitatives Verfahren gewählt, da es darum geht, die Objekte in Kategorien einzuteilen. Im zweiten Teil wird das Modellieren mit Tools im Rahmen von TechKreativ-Workshops untersucht. Diese können als künstliche Umgebungen angesehen werden im Gegensatz zu Situationen, in denen Maker_innen im Privaten modellieren. Da solche Situationen schwer zugänglich für Beobachtungen sind, wurde die beschriebene Rahmung bevorzugt. Durch die freie Gestaltung hinsichtlich des Arbeitsablaufs und der selbstbestimmten Aufgabenstellung und deren Bearbeitung durch die Modellierenden sollte gewährleistet werden, dass diese Konstellation nicht allzu laborartig und künstlich erscheint. Ähnliche Situationen sind auch an Orten, an denen sich Maker_innen zum gemeinsamen Basteln treffen, wie z.B. in FabLabs, vorstellbar.

7.3 Modelle aus Makerworkshops – eine Analyse

7.3.1 Datenerhebung

Um zu untersuchen, welche Arten von Modellen in Workshops mit den existierenden Tools und analogen Methoden erstellt werden, wurden dort dokumentierte Modelle analysiert. Dazu wurde auf Datenmaterial zugegriffen, das in TechKreativ-Workshops seit 2008 erhoben worden war und das in digitalisierter oder materieller Form aufbewahrt worden war. Diese Dokumentationen waren von Workshopteilnehmenden erstellt worden – vor, während oder nach dem Konstruieren der Artefakte.

Dieses Datenmaterial wurde vor dem Hintergrund gewählt, dass es verfügbar war, einige der Modellierungstätigkeiten bereits als Teil dieser Arbeit konzipiert wurden und die Workshops von Tutor_innen begleitet wurden, zu denen die Autorin zum Zeitpunkt der Analyse noch in Kontakt stand[2]. So war es möglich, mehr über Rahmenbedingungen und Kontextfaktoren zu erfahren und dadurch das Material besser einschätzen zu können.

[2] Auch sie selbst war an einigen der Workshops als Tutorin beteiligt.

7.3.2 Auswertung der Modelle mittels Inhaltsanalyse

In Anlehnung an die skalierende strukturierende Inhaltsanalyse nach Mayring (2010, S. 102) wurde ein Vorgehensmuster entwickelt (siehe Abbildung 7.1 links). Die folgenden Abschnitte legen die getätigten Analyseschritte in chronologischer Reihenfolge dar.

Analyse des Ausgangsmaterials

Beim zu analysierenden Material handelt es sich um Fallbeispiele, die im Rahmen von TechKreativ-Workshops entstanden sind. Das Material umfasst von Teilnehmenden digital oder analog erstellte Dokumente in Form von Amici-Programmcode, Einträgen in das VirtualLab, Skizzen und Einträgen in MoLab. Diese wurden nach Verfügbarkeit gewählt und sind nicht repräsentativ. Die Workshopteilnehmenden wurden in den meisten Fällen durch Ausschreibungen gewonnen. Bei den Artefakten handelt es sich um Ergebnisse aus Gruppenarbeit, d.h. um gemeinschaftlich erstellte Artefakte. Es ist nicht auszuschließen, dass Eingriffe der Workshoptutor_innen mitunter das Material beeinflusst haben. Die vorliegenden Programme wurden in den Workshops erstellt mit dem Zweck, die dort gebauten Artefakte zu programmieren. Modelle wie VirtualLab-Einträge, Skizzen und MoLab-Einträge waren dagegen zur Artefaktkonstruktion nicht unbedingt notwendig. Die Teilnehmenden wurden von den Tutor_innen dazu angeregt.

Bei mit Amici erstellten Modellen handelt es sich um graphischen Programmcode, also eine eigene visuelle Sprache, die als Abbildungen vorlagen. Die übrigen Modelle umfassen Texte, Zeichnungen, Fotografien und Mischformen in Form von Websites (MoLab und VirtualLab).

Richtung der Analyse und Fragestellung

Mit der Analyse sollen Charakteristika der Modelle erschlossen werden und entsprechend der Modellperspektiven und -kategorien eingeordnet werden. Die zu untersuchende Frage (siehe Abschnitt 7.1) lautet: Was kennzeichnet die in TechKreativ-Workshops dokumentierten Modelle, welche der genannten Perspektiven enthalten diese und wie sind die Modelle nach AMT einzuordnen? Die Analyse orientiert sich an der AMT und geht von den in Abschnitt 6.1.3 abgeleiteten Perspektiven für das Erstellen stofflich-digitaler Artefakte aus. Die Analyse ist offen für neue Beobachtungen.

Analyseablauf

Neben Texten sollen Programme und Bilder analysiert werden sowie Webseiten, die wiederum die zuvor genannten Elemente enthalten können. Als Analyseeinheiten wurden festgelegt:

- Bilder: Erkennbare Details bis Gesamtbild
- Text: Präposition bis Gesamttext als Einheit
- Programm: Einzelblock und dessen Text oder Bild (bei Amici-Programmen) oder Text (bei Arduino-Code) bis hin zum Programm als Ganzem
- Webseiten: Einzelne Elemente (Bilder, Text usw.) und als Ganzes

Es wurden Kategorien gemäß der theoretischen Erkenntnisse dieser Arbeit mit Definition, Ankerbeispielen und Kodierregeln aufgestellt (ausführliche Beschreibung in Tabelle A im Anhang). Als Kategorien gelten die Perspektiven Programmierung, Hardwarekonfiguration, Artefaktkorpus, interaktives Verhalten[3] und pragmatische Merkmale, sowie die Modellkategorien graphische Modelle, textuelle Modelle und technische Modelle (jeweils mit Unterkategorien) aus Kapitel 3, Abbildung 3.1. Die pragmatischen Merkmale wurde in die Unterkategorien Kontext, Intention und Modellsubjekt gegliedert, um diese eindeutiger identifizieren und unterscheiden zu können.

Für die Auswertung wurden im vorliegenden Datenmaterial Elemente in Abbildungen oder ganze Bilder sowie Textbausteine anhand des Kategorienschemas farblich codiert. Das jeweilige Vorkommen der Modelleigenschaften in den einzelnen Modellen wurde quantitativ nach Häufigkeit zusammengefasst. Danach wurde ausgewertet, welche Perspektiven mit welchen Modellarten und mit welchen Modellierungstechniken abgebildet wurden.

Aufgrund des Umfangs werden die Ergebnisse in einem eigenen Abschnitt dargestellt und besprochen. Eine Zusammenfassung der quantitativen Verteilung auf Modellperspektiven, geordnet nach Tools und nach Modellarten, ist auf Seite 158 in Abbildung 7.15 dargestellt.

7.3.3 Ausführliche Darstellung der Ergebnisse der Modellanalyse

Im Folgenden werden die Ergebnisse geordnet nach den Modellierungsmethoden und -tools, mit denen sie erstellt wurden, vorgestellt.

Modelle vorgespielter Szenarien

In Abschnitt 6.3.2 wurde eine eher ungewöhnliche Modellierungsmethode des körperlichen und gegenständlichen Vorspielens vorgestellt, die in einem TechKreativ-Workshop zur Ideenfindung eingesetzt wurde. Nach dem die Methode eingeführt worden war, entwickelten die einzelnen Gruppen zunächst mehrere Ideen, die sie mit einem Fotoapparat und handschriftlichen Notizen festhielten, entschieden sich aber bald für eine Idee, mit der sie sich ausgiebiger beschäftigten. Alle Projektgruppen bildeten ihre vorgespielten Szenarien in Text und Bild ab, indem je ein Gruppenmitglied

[3] Abgekürzt als „Programm“, „Hardware“, „Korpus“, „Interaktion“

die bzw. den Vormachende_n fotografierte und Notizen zu den Ideen aufschrieb. Anschließend wurden diese von einer Tutorin abgetippt und mit den Fotos auf einer Seite digital zusammengestellt, ausgedruckt und den Gruppen ausgehändigt.

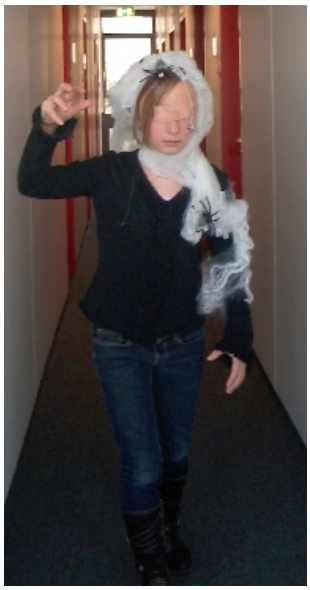
Analysiert wurde von jeder Projektgruppe die Dokumentation des Szenarios, das sie den anderen Teilnehmenden als ihre Projektidee vorstellte. Insgesamt lagen so fünf Dokumentationen für die Analyse vor. Fotos, die redundante Situationen zeigten, wurden weggelassen.

Die vorliegenden Modelle der Ideen bestanden aus graphischen und textuellen umgangssprachlichen Modellen. Als ein Modell wird jeweils ein Foto mit dazugehörigen Notizen betrachtet. Bildfolgen wurden nicht explizit erstellt. Alle Bilder zeigen eine Idee für den Korpus des zu modellierenden Artefakts. Betrachtet man die Modelle, so wird durch Andeutungen auf Fotos, z.B. durch eine Handbewegung, in drei Fällen auch das dynamische Verhalten des Artefakts angedeutet. Bei vier der Dokumentationen wird das Verhalten des Artefakts im Text stichwortartig beschrieben (siehe Abbildung 7.2). Informationen über Hardwareelemente und erste algorithmische Beschreibungen finden sich in einem Projekt, dessen Teilnehmer Vorerfahrung mit Arduino-Technologien hatten.

Intentionen, also warum und wozu das Projekt erstellt wird, wurden nicht abgebildet. Kontexte sind in zwei Fällen ansatzweise erkennbar, indem das erdachte Artefakt in vorhandene Möbelstücke integriert ist oder am Körper getragen wird. Subjekte werden durch die vorspielenden Personen implizit abgebildet und wurden deshalb nicht gewertet. Die quantitativen Ergebnisse sind in Abbildung 7.3 dargestellt.

Insgesamt ergab die Analyse, dass diese Methode sich eignet, Ideen durch gegenständliches Vormachen und bildliches und textuelles Dokumentieren zu modellieren. Dabei fokussierten die Darstellungen der Ideen auf die äußere Gestalt des Artefaktkorpus. Interaktionen mit dem Artefakt wurde nur in wenigen Bildern angedeutet und es wurden keine Bildfolgen aufgenommen, um einen Interaktionsverlauf zu dokumentieren. Jedoch wird in den Texten meist auch das (interaktive) Verhalten – auf unformale Weise – beschrieben. Diese Art der Modellierung scheint sich folglich insbesondere für die Entwicklung des statischen Artefaktkorpus und auch bedingt für dessen dynamisches Verhalten in einer frühen Projektphase zu eignen.

Ergänzend sei gesagt, dass die meisten Gruppen ihre Ideen später beibehielten und die finalen Artefakte große inhaltliche Ähnlichkeit bezogen auf äußere Gestalt und Verhalten zeigten, obwohl diese Methode in einer frühen Phase des Workshops eingesetzt wurde, als die zur Verfügung stehenden Technologieteile noch nicht bekannt waren. Ein Beobachtungsprotokoll und ein anschließendes Auswertungsgespräch der beteiligten Tutor_innen (die jedoch nicht systematisch ausgewertet wurden) lässt darauf schließen, dass dadurch die Ideen im Voraus besser durchdacht wurden, d.h. Wider-



Aus Hut kommen Spinnen / Wasser.

Abbildung 7.2: Beispiel: Dokumentation der Projektidee für einen ‚spukenden' Hut auf Foto mit zugehöriger Notiz (Text im Original handschriftlich)

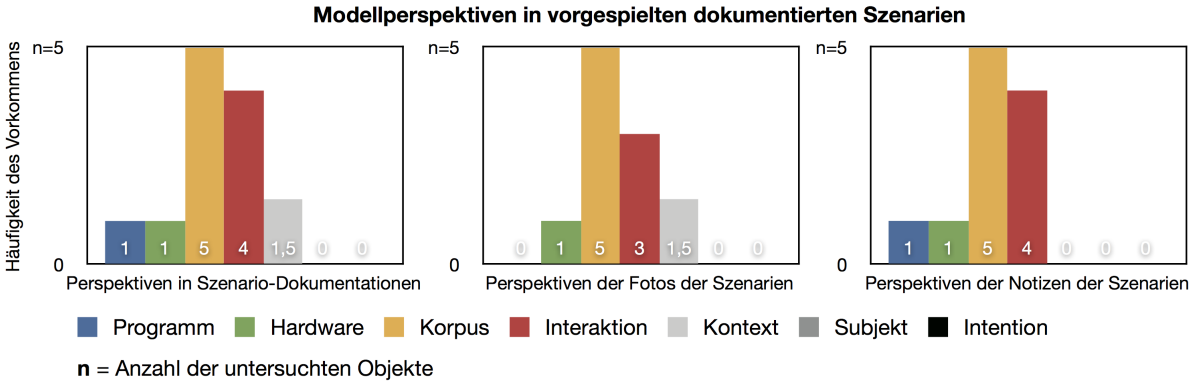


Abbildung 7.3: Quantitative Ergebnisse: Vorkommen verschiedener Modellperspektiven und Modellarten in dokumentierten Szenarien

Workshop: Name Jahr; Dauer; Thema (Technologie)	**Demographie**	**Art der Skizzen: Vorgaben; Zweck**	**Anzahl Skizzen**
„Smart Fashion“ 2008, 5 Tage; „Intelligente Kleidung von morgen aus Kleidung von gestern“ (LilyPad Arduino)	10 - 14 Jahre, 15 Teilnehmende (13 w, 2 m), ohne Vorerfahrung	Umrisse von Kleidungsstücken, in die Hardwarekomponenten u.ä. eingezeichnet werden konnte; Umsetzung von Ideen konkretisieren	6
„Robot DIY“ 2011; 2 Tage; „Robots selbst bauen und programmieren“ (Lego Mindstorms)	21 - 29 Jahre, 6 Teilnehmende (5 m, 1 w), Studierende der Soziologie, Informatik und Ingenieurwesen	leeres Papier; zu Ideenfindung: „Was würdet ihr erfinden, wenn alles möglich wäre?“ (laut Ablaufplan vor Materialeinführung).	8
„Wear & Move“ 2009; 5 Tage; „Intelligent garments (...) that can capture movements“ (LilyPad Arduino)	11 - 14 Jahre, 7 Teilnehmende (4 m, 3 w), teilweise mit Vorerfahrung aus anderen Workshops	Am Ende des ersten Tages (nach Ideenfindung): Formular mit Aufforderung „Draw a sketch of your groups‘ project idea!“	10

Tabelle 7.1: Rahmenbedingungen der drei Workshops und Zweck der Skizzen

sprüche oder Unstimmigkeiten bemerkt und korrigiert wurden.[4] Zur Ideenfindung, -ausgestaltung und -dokumentation scheint es anregend zu sein, sich mit dem Material (ggf. auch Umfeld oder Bewegung) körperlich auseinandersetzen zu können. Auch dynamische Eigenschaften konnten damit ansatzweise modelliert werden.[5] Das Spielen von Szenarien wird daher gemeinsam mit der Möglichkeit, dies durch Fotos als spätere Orientierungshilfe festzuhalten, als eine relevante unformale Modellierungsmethode erachtet, um erste Vorstellungen des Artefakts und der Interaktion mit diesem zu dokumentieren.

Skizzen als Modelle

In TechKreativ-Workshops besteht die Gelegenheit, auf Papier Notizen zu erstellen oder Skizzen zu zeichnen. Diese können dabei helfen, Projektideen festzuhalten und ggf. zu spezifizieren. Meist wird davon wenig Gebrauch gemacht. Um Skizzen zu evozieren, wurden in zwei Workshops Vorlagen zur Verfügung gestellt, die schon Umrisse eines Kleidungsstücks (siehe Abschnitt 6.3.1) zeigten bzw. eine Aufforderung enthielten.

Der Übersichtlichkeit halber sind die Workshops, die jeweiligen Skizzentypen, die Anzahl der untersuchten Skizzen[6] und die Rahmenbedingungen in Tabelle 7.1 gelistet.

[4] Dies wird daran festgemacht, dass fast alle umgesetzten Projekte auch den ursprünglichen Ideen entsprachen, was in Workshops mit bloßem Brainstorming nach Beobachtungen nicht der Fall war.

[5] Zu bemerken wäre noch, dass eine Gruppe spontan versuchte mit der Videofunktion des Fotoapparates ihre Idee zu filmen. Aufgrund zu dunkler Lichtverhältnisse gaben sie diesen Versuch wieder auf. Für zukünftige Einsätze der Methode sollte die Möglichkeit der Videodokumentation bewusst angeboten werden, um die Eignung von Video für das Dokumentieren von interaktivem Verhalten zu untersuchen.

[6] Alle Skizzen, die zum Zeitpunkt der Analyse auffindbar waren, wurden ausgewertet.

Im Workshop „Robot DIY“[7] wurden Skizzen zur Ausgestaltung von Ideen eingesetzt. Von den acht untersuchten Skizzen zeigen nur ca. die Hälfte formal beschreibbare statische Charakteristika (Hardwareelemente) – trotz der größtenteils technischen Studienfächer der Teilnehmenden. Alle zeigen das Produkt, ein Großteil auch die Interaktion, also dynamisches unformales Verhalten, wenige etwas über Kontext und Intention. Die meisten Dokumente zeigen nicht nur bildliche Motive, sondern ergänzen diese um verhältnismäßig ausführliche Beschreibungen (siehe Abbildung 7.4).

Die zehn Skizzen des „Wear & Move“-Workshops wurden gezeichnet, um die technische Umsetzung der Projektidee zu konkretisieren (siehe Abbildung 7.5). Alle bilden die äußere Gestalt des Artefaktkorpus und die Hardware ab, z.T. wurden auch Details des Artefaktkörpers eingezeichnet. Über geplantes interaktives Verhalten und Intentionen wird nur vereinzelt Auskunft gegeben. Einige Skizzen beinhalten textuelle Modelle, um Teile des Gezeichneten zu erläutern.

Im Workshop „Smart Fashion“[8] wurden Skizzen erstellt, um die Integration der Hardwareelemente in das Kleidungsstück (vorgezeichnete Kleidungsumrisse) zu konzipieren (siehe Abbildung 7.6). Dementsprechend bilden alle Dokumente die äußere Gestalt des Kleidungsstücks (durch die Vorlage vorgegeben) und die Hardware ab, z.T. werden auch Details des Artefakts eingezeichnet. Das gewünschte interaktive Verhalten und Intentionen deuten nur wenige an.[9]

Zusammenfassung der Analyse von Skizzen

Die untersuchten Skizzen zeigen alle im Wesentlichen den zu modellierenden Artefaktkorpus (in skalierter, aber Proportionen erhaltender Form) mit gestaltenden Attributen. Dreiviertel der Skizzen zeigen auch Hardwareelemente, aber nicht unbedingt die notwendige Konfiguration, und modellieren dadurch auch die Schnittstelle zwischen dem Artefaktkorpus und den Hardwarekomponenten. Ein Drittel zeigt Hinweise auf interaktives Verhalten. Lediglich drei bilden Intentionen ab, indem durch Text der Zweck des Artefakts festgehalten wird. Die Skizzen, auf denen bereits Umrisse von Kleidungsstücken vorgezeichnet waren, enthalten mit einer Ausnahme im Vergleich zu den freien Skizzen nur Attribute der Perspektiven Hardware und Artefaktkorpus. Generell wurden in ca. ein Drittel bis der Hälfte der Skizzen Bildmodelle durch textuelle Modelle ergänzt. Verhältnismäßig häufig fanden sich in Texten eine Interaktionsperspektive oder pragmatische Merkmale. Meist wurden textuelle Modelle benutzt, um

[7] Der Workshop wurde im Rahmen des Forschungsprojekts Skudi (`http//www.skudi.org`, aufgerufen am 14.05.2014) veranstaltet.

[8] Der Workshop wurde im Rahmen des Forschungsprojekts EduWear (`http://dimeb.de/eduwear/about-2/`, aufgerufen am 14.05.2014) veranstaltet.

[9] Zur praktischen Eignung dieser Methode sei noch bemerkt, dass es sich als nachteilig erwies, dass die Skizzen nicht haptisch und dreidimensional genug waren, um auch den Verlauf der Leitungen z.B. an der Seite des Kleidungsstücks abzubilden. (Diese Aussage beruht auf protokollierten Beobachtungen der Tutorinnen während des Workshops.)

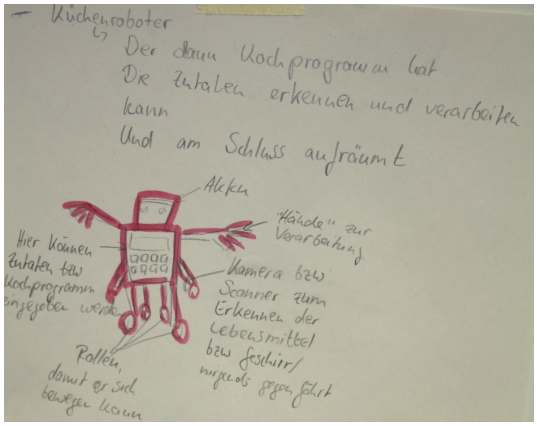


Abbildung 7.4: Beispiel: Skizze für Küchenroboter aus dem Workshop „Robot DIY“

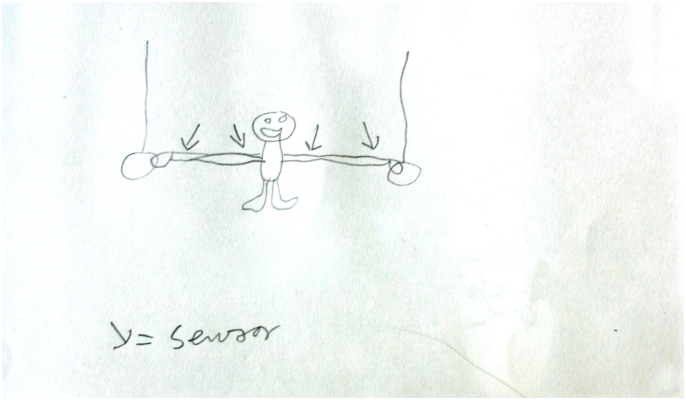


Abbildung 7.5: Beispiel: Skizze für Messgerät beim Turnen aus dem Workshop „Wear & Move“

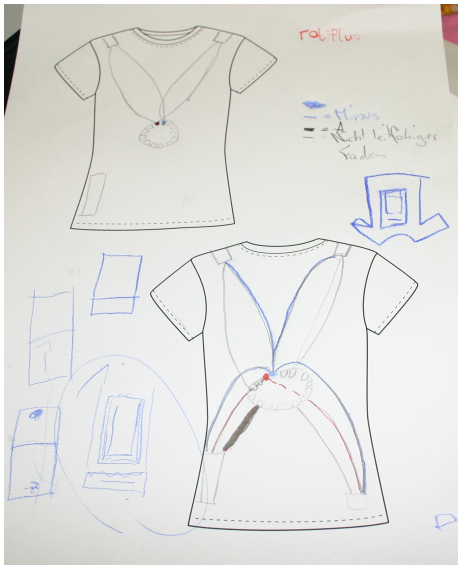


Abbildung 7.6: Beispiel: Skizze für Massage-T-Shirt aus dem Workshop „Smart Fashion“

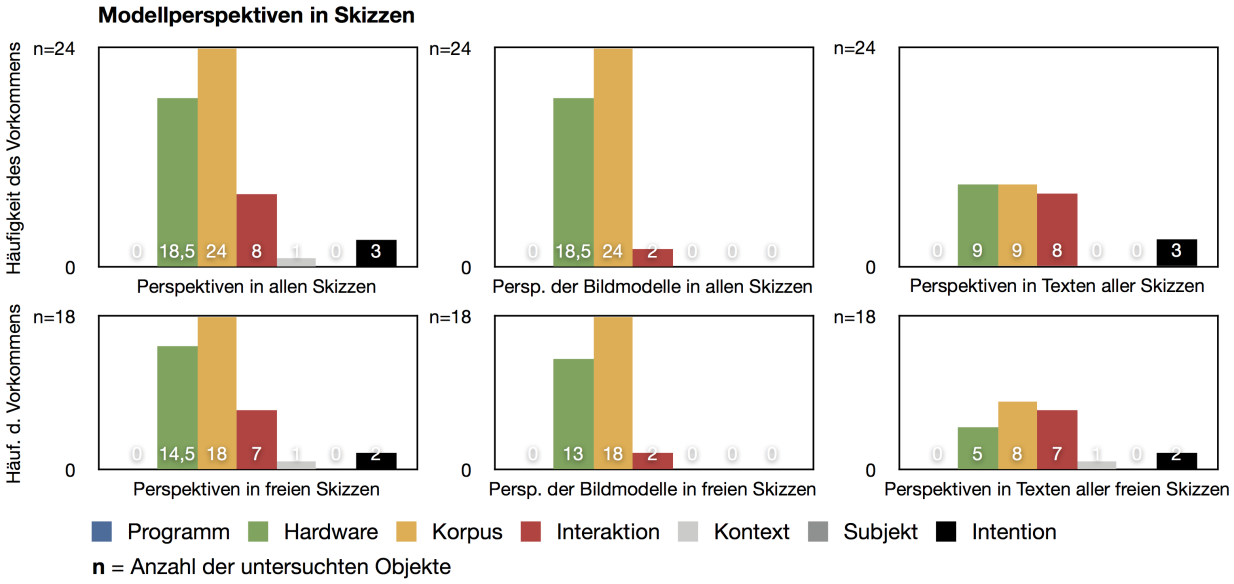


Abbildung 7.7: Quantitative Ergebnisse: Vorkommen verschiedener Modellperspektiven und Modellarten in Skizzen

Dinge, die so auf dem Bild nicht differenzierbar waren, z.B. Sensoren, genau zu benennen.

Bei der quantitativen Zusammenfassung der Ergebnisse muss beachtet werden, dass die drei Skizzentypen zu unterschiedlichen Zwecken erstellt wurden. Daher können die Auswertungen nur Tendenzen aufzeigen, welche Eigenschaften mit Skizzen wie abgebildet werden, bzw. welche nicht abgebildet werden. Die Kleidungsskizzen wurden mit Vorlagen erstellt, in die schon eine Artefaktkorpusperspektive eingezeichnet war. In Abbildung 7.7 werden deshalb zusätzlich auch die Ergebnisse ohne Berücksichtigung der Kleidungsskizzen angegeben.

Visuelle Programme als Modelle

Ein Amici-Programm ist optionaler Teil einer Projektdokumentation im VirtualLab und in MoLab. Aus konzeptioneller Sicht zeigt Amici-Code mit der Programmierung und den Hardwareanschlüssen nur einen bestimmten Ausschnitt des Artefakts und nicht das vollständige Projekt. Um zunächst herauszufinden, welche Informationen über das Artefakt in diesen Modellen enthalten sind, wurden Amici-Programme, die als Teil einer Projektdokumentation hochgeladen worden waren, eigenständig untersucht.

Auf die besondere Rolle der Programmierung im Modellierungsprozess wurde bereits mehrfach hingewiesen (z.B. in Abschnitt 3.2.3). Sie ist ein, wenn nicht *der* wesentliche Teil des Modellierungsprozesses, in dem das beabsichtigte interaktive Verhalten endgültig formalisiert (und algorithmisiert) werden muss. In TechKreativ-Workshops, in denen mit Arduino-Technologie gearbeitet wird, wird zur Programmierung der

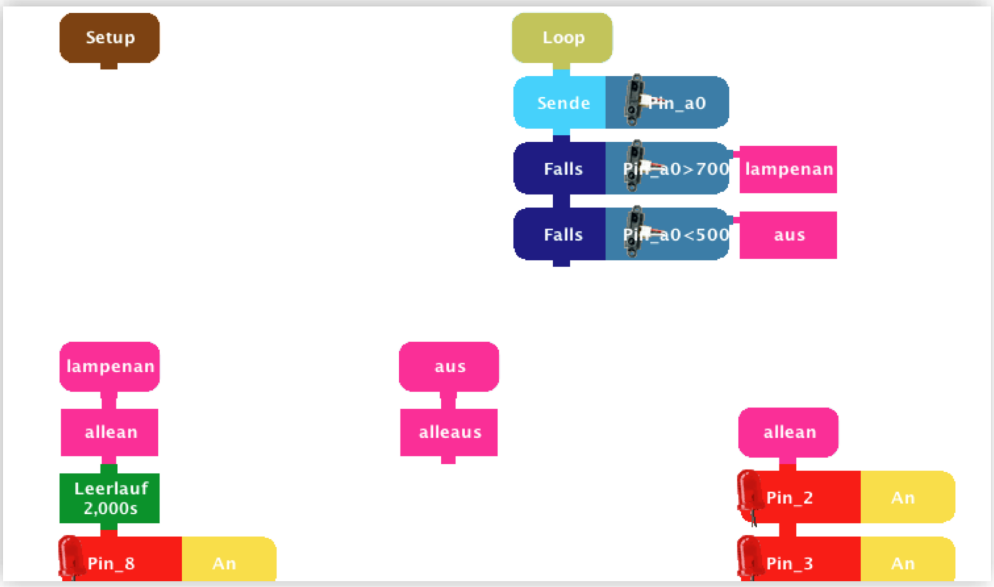


Abbildung 7.8: Beispiel: Ausschnitt eines Amici-Programms mit Methoden

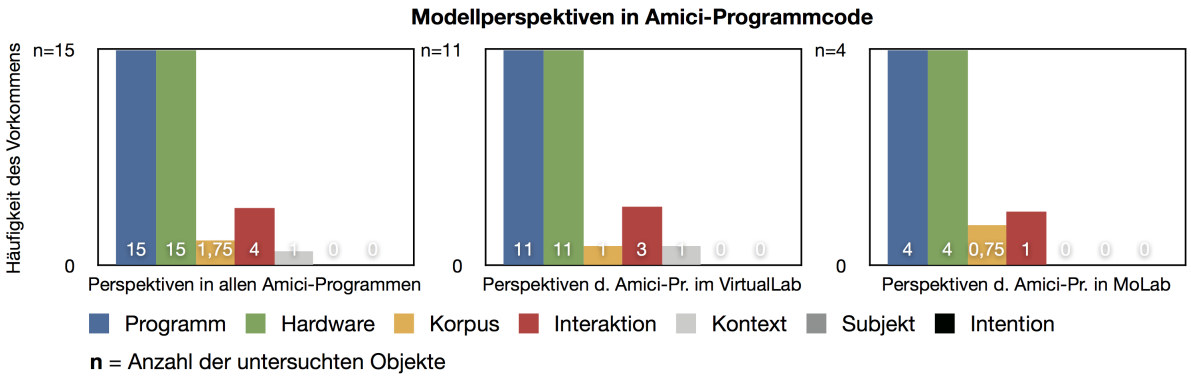


Abbildung 7.9: Quantitative Ergebnisse: Vorkommen verschiedener Modellperspektiven und Modellarten in visuellen Amici-Programmen

Controller die Visuelle Programmierumgebung Amici eingesetzt, die in Abschnitt 5.3.5 eingeführt wurde. In Abschnitt 6.4.2 wurde dargestellt, wie Amici umgestaltet wurde, um besser an die Hardware und an das Artefakt, das nebenbei konstruiert wird, anzuknüpfen. In diesem Abschnitt sollen, wie oben bei anderen Arten von Modellen geschehen, mit Amici erstellte Programme hinsichtlich ihrer Modellattribute untersucht werden. Die insgesamt 15 analysierten Programme wurden Projektdokumentationen entnommen, die mit dem VirtualLab und mit MoLab erstellt worden waren und die Amici-Programmdateien als Anhang enthielten.

Die Analyse ergab, dass die untersuchten Amici-Programme Attribute des Verhaltens des Artefakts in Form eines programmierten Algorithmus, der durch Blöcke dargestellt wird, enthalten. Auch finden sich, obwohl es sich vorwiegend um eine Programmiersprache handelt, umfassende Informationen zur Hardwarekonfiguration. Diese geben vorwiegend an, welche Sensoren und Aktuatoren an welchen Pinns des Boards angeschlossen sind.

Neben den Abbildungen der Blöcke und Sensor- bzw. Aktuator-Icons, die von Amici vorgegeben werden und so konzipiert wurden (siehe Abschnitt 6.4.2), fanden sich in der Analyse ansatzweise auch als unformal eingestufte Perspektiven. So geben die selbst gewählten Methodennamen Einblick in die Gestalt des Artefaktkorpus oder dessen interaktives Verhalten. Von den analysierten Amici-Programmen des VirtualLab enthielt etwa jedes vierte Methoden.[10] Die Methodennamen enthalten Attribute, die den Kontext des Artefakts und sein Aussehen und die mögliche Interaktion minimal abbilden. Beispiele sind die Farbenbezeichnungen von LEDs („blau“, „lila“), oder „Zukunft“, „Zukunft2“ und „Zukunft3“ bei einem Orakel, das damit über das intendierte Verhalten des Artefakts und dessen Einbettung in einen Kontext etwas aussagt. Namen wie „allean“, „alleaus“ „lampean“ geben auch Informationen über das Produkt (enthält Lichtquellen) und das Verhalten (Licht wird an- und ausgeschaltet) (siehe Abbildung 7.8). Wobei Bezeichnungen wie „allean“ oder „alleaus“ erst in Verbindung mit den Hardware-Symbolen (in diesem Fall LEDs) erkennen lassen, was hier angeschaltet wird.

In der Zusammenfassung enthalten Programme, die mit Amici erstellt wurden, nicht nur Informationen über einen Algorithmus, sondern auch wesentliche Informationen über die Hardwarekonfiguration und überraschenderweise auch Attribute der unformalen Perspektiven des Artefakts. Intentionen oder Attribute, die den sozialen Kontext abbilden, sind aber nicht zu finden.

Projektdokumentationen im VirtualLab

Untersucht wurden Projektdokumentationen, die mit dem VirtualLab (siehe Abschnitt 5.1.2) in zwei Workshops zum Thema „Magisch, mysteriös, zauberhaft – Im Wunder-

[10] In MoLab enthielten alle Amici-Programme Methoden, hier handelte es sich aber um verschiedene Versionen desselben Projekts.

land von morgen“[11] (2011-2012) erstellt wurden. Die Workshops dauerten fünf bzw. vier Tage, Ziel war es laut Ausschreibung „magische oder mysteriöse Dinge zu entwerfen und zu konstruieren“. Die insgesamt 25 (12 w, 13 m) Teilnehmenden waren zwischen 9 und 14 Jahren alt. Die Dokumentationen entstanden am Ende der Workshops aus dem Auftrag, das Projekt in das VirtualLab einzutragen.

Ausgewertet wurden elf Projekte, die im VirtualLab dokumentiert wurden. Die analysierten Dokumentationen waren jeweils am Ende eines Workshops erstellt worden. Somit sind sie als abbildende Modelle der (vorläufig) fertigen Artefakte anzusehen. Die Projektdokumentationen enthalten zumeist eine textuelle Beschreibung, ein Foto, Programmcode, Autor_innennamen und Information über verwendete Hardwareelemente. Diese Elemente sind im VirtualLab als Eintragungsfelder vorgegeben, wovon Titel und Beschreibung Pflichtfelder sind und der Autorenname automatisch ausgefüllt wird. Eine solche Dokumentation eines Projekts wird hier als Artefakt-Modell bezeichnet und entsprich einem ‚Eintrag‘ im VirtualLab. Ein Artefakt-Modell enthält Modelle einzelner Aspekte oder Teilsysteme, die gemeinsam als ein umfassendes Modell das Artefakt (möglichst) vollständig aus verschiedenen Perspektiven abbilden können. Hier wurden sowohl die einzelnen Modelle (Texte, Bilder usw.) analysiert als auch der gesamte VirtualLab-Eintrag als Artefakt-Modell.

Die Analyse ergab, dass Fotos des Artefakts in allen Fällen den Artefaktkorpus als fotografisches, ikonisches Bildmodell abbilden. Wenige lassen auch das dynamische Verhalten oder Kontext- und Modellsubjekt-Informationen erkennen. Texte bilden hauptsächlich die weitgehend unformal beschriebene Interaktion und das Aussehen des Artefakts ab sowie den Kontext, in den das Artefakt eingebettet ist oder dem es entstammt (zumeist den Zusammenhang zum Workshopthema). In wenigen Fällen finden sich formal anmutende, algorithmische Beschreibungen (z.B. durch Wertangaben statt Eigenschaften wie „dunkel“). Hardwareteile werden oft benannt, aber die genaue Konfiguration (z.B. der Pinnanschluss) nicht. Fast Dreiviertel der Autoren thematisieren sich selbst als Modellierende. Die Intention, warum und zu welchem Zweck das Artefakt erstellt wurde, erwähnen nur knapp ein Fünftel (siehe Abbildung 7.11).

Alle Artefakt-Modelle geben Auskunft über Hardware und Programmierung anhand des beigefügten Amici-Codes (siehe oben). Darüber enthalten sie die Autorennamen der Modellsubjekte sowie (sofern vorhanden) die verwendeten Hardware-Bauteile. Über die Fotos und teilweise auch die Texte enthalten sie Attribute der Gestalt des Artefaktkorpus. Eine Perspektive auf die Interaktion ist ebenfalls in fast allen, pragmatische Merkmale in gut der Hälfte der untersuchten Projektdokumentationen erkennbar.

Im Durchschnitt werden in einer Projektdokumentation (d.h. einem VirtualLab-Eintrag), die ein Artefakt-Modell darstellt, nahezu alle Perspektiven abgebildet, die

[11] Der Workshop wurde im Rahmen des Forschungsprojekts informAttraktiv (`http://www.dimeb.de/informattraktiv`, aufgerufen am 14.05.2014) veranstaltet.

zum Nachvollziehen oder Wiederaufnehmen des Projekts notwendig erscheinen. Das Format des VirtualLab scheint das Dokumentieren umfangreich zu unterstützen. Auch Subjekt- und Kontextaspekte wurden bei ca. Zweidritteln der Projektdokumentationen erwähnt. Intentionen werden jedoch in der Mehrzahl nicht genannt. Die Artefakt-Modelle können durchaus als Vorbilder für andere Projekte dienen. Sie sagen aber wenig über den Prozess aus, der auch Probleme und Misserfolge enthalten mag. Nur in manchen Beschreibungen finden sich Hinweise. So wird z.B. in einer Beschreibung bemerkt, man brauche viel Geduld für das Projekt.

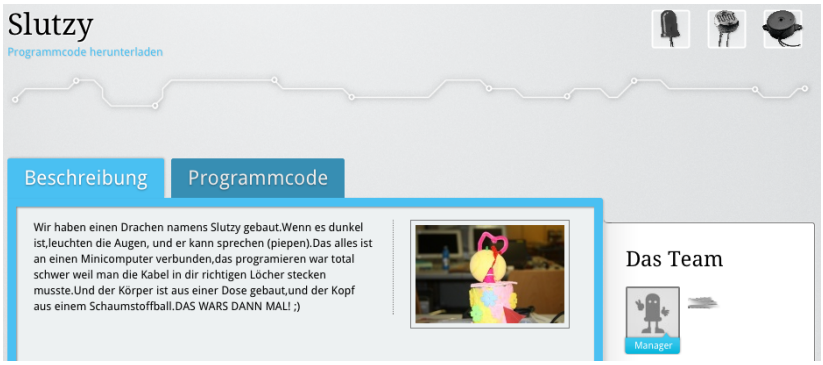


Abbildung 7.10: Beispiel: Projektdokumentation im VirtualLab

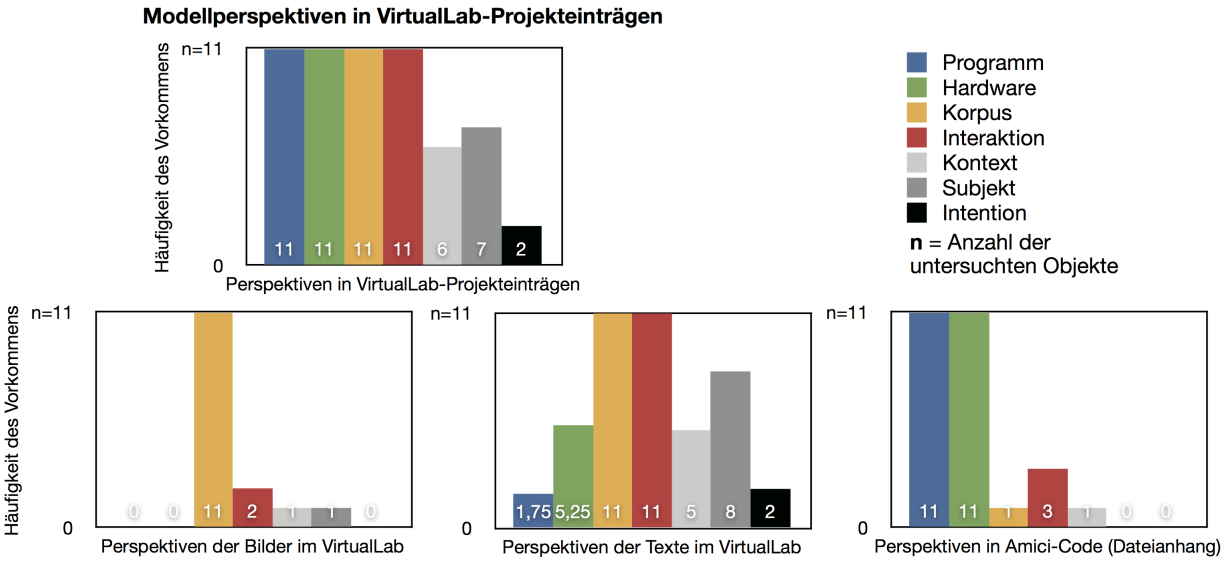


Abbildung 7.11: Quantitative Ergebnisse: Vorkommen verschiedener Modellperspektiven und Modellarten in VirtualLab-Einträgen

Modelle aus MoLab

Mit der Webplattform MoLab (siehe Abschnitt 6.5) können Projekte durch viele einzelne Modelle dokumentiert werden, welche verschiedene Perspektiven auf das Artefakt einnehmen können. Die Modelle können so gruppiert werden, dass sie zusammengenommen ein Modell des Artefakts bzw. Projekts und/oder des Modellierungsprozesses bilden. Die Modellanalyse untersuchte Projekteinträge aus MoLab (siehe Abschnitt 6.5), die im Rahmen eines wöchentlich stattfindenden Workshops mit fünf Terminen (siehe Abschnitt 7.4) und in einem späteren, durchgängigen viertägigen Workshop von Jugendlichen erstellt worden waren. Da im späteren Workshop eine überarbeitete Version von MoLab (bezeichnet als 2.0) eingesetzt wurde, werden die jeweiligen Ergebnisse getrennt vorgestellt.

MoLab Version 1.0 (Fallstudie): Die MoLab-Einträge, die während der Fallstudie erstellt wurden, enthalten meist Amici-Programmcode und/oder Fotos, die den aktuellen Zustand des Artefakts oder die Projektautor_innen (Modellsubjekte) zeigen. Die vorliegenden Einträge stellen jeweils den aktuellen Status des Projekts dar, das im jeweils nächsten Eintrag um neue Funktionen erweitert worden war, z.B. weitere Farben der LEDs, Einbauen des Sensors, usw. Die einzelnen MoLab-Einträge enthalten Attribute von Hardware und Programm, die hauptsächlich durch den beigefügten Amici-Code abgebildet werden (siehe oben). Auch ist die Hardwarekonfiguration auf einigen Fotos der zusammengesteckten Hardwarekomponenten zu sehen (siehe Abbildung 7.12). Betrachtet man alle Modelle auf dem Modelboard als *ein* Modell des Modellierungsprozesses, so bildet dieses Modell die Perspektiven Hardware, Programm, Artefaktkorpus und interaktives Verhalten ansatzweise ab. Intentionen als pragmatische Eigenschaften werden nicht aufgeführt. Die Modellierenden bildeten sich aber oft selbst ab. Es schien ihnen wichtig zu sein, sich als Personen darzustellen und als Autor_innen und Modellsubjekte in Erscheinung zu treten. Während Hardware und Programmierung im Amici-Code und als Hardwarekonfiguration auf Fotos abgebildet wurden, werden Interaktionsbeschreibungen nur indirekt mitgeteilt, z.B. im Titel, Dateinamen des Programms und durch Methodennamen in Amici. Sie sind aber nicht explizit beschrieben. Der Artefaktkorpus wurde insgesamt auf einer Skizze und auf einem Foto abgebildet.

Insgesamt wurden Modelle aus fast allen Perspektiven abgebildet mit unterschiedlichen Schwerpunkten und in verschiedenen Phasen des Projekts. Überwiegend wurden Bildmodelle und Programmcode zur Dokumentation genutzt. Mit Texten wurden bruchstückhafte Attribute der Hardware und des interaktiven Verhaltens abgebildet. Der Hardwarekonfigurationen, dem Korpus sowie Modellsubjekten zugeordnete Attribute des Artefakts wurden in Bildmodellen abgebildet. Hardwarekonfiguration, Programm und angedeutete Informationen zu Interaktion und Artefaktkorpus wurden im Amici-Programmcode festgehalten.

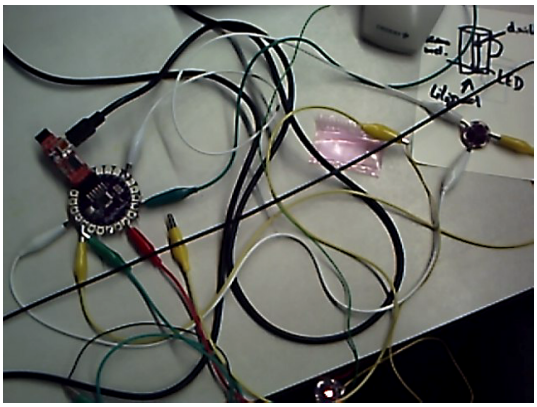


Abbildung 7.12: Foto zusammengesteckter Hardwarekomponenten

Abbildung 7.13: Foto eines fertigen Artefakts ‚in Aktion'

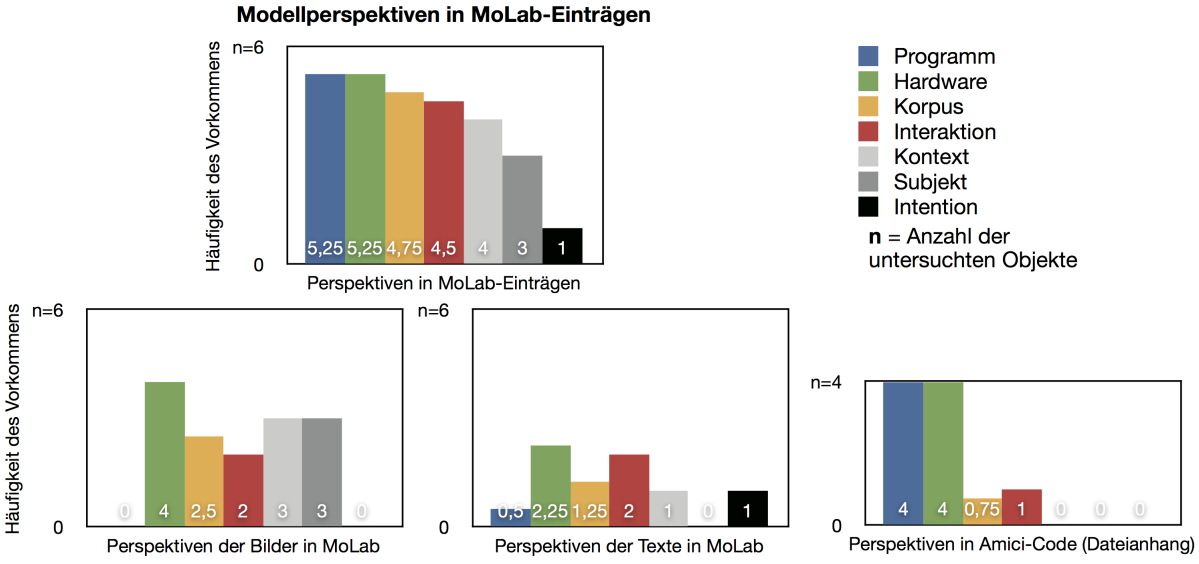


Abbildung 7.14: Quantitative Ergebnisse: Vorkommen verschiedener Modellperspektiven und Modellarten in MoLab-Einträgen

MoLab Version 2.0: MoLab wurde (in überarbeiteter Version, siehe Abschnitt 7.4.6) auch in zwei parallel stattfindenden viertägigen TechKreativ-Workshop eingesetzt, an denen ca. 30 Jugendliche im Alter von 13 bis 18 Jahren teilnahmen. Genutzt wurde MoLab nur von zwei Projektgruppen (im Folgenden Projektgruppe I und Projektgruppe II genannt).

Das in MoLab erstellte Projekt von Projektgruppe I enthält eine Beschreibung ihrer Projektidee. Dieses MoLab-Modell besteht aus einem MoLab-Eintrag, der nur Text enthält. Es beschreibt das Projekt, eingebettet in soziale Kontexte, indem es die Situation und das soziale Gefüge beschreibt und die dort stattfindende Interaktion und den Zweck.[12] Verbaute Sensoren und Aktuatoren werden benannt. Eine algorithmische Beschreibung oder eine genaue Spezifikation der Hardware findet sich in der Beschreibung nicht, obwohl sie am dritten Workshoptag erstellt wurde. Diese Dokumentation zeigt daher hauptsächlich das Produkt und die Interaktion und erwähnt den sozialen Kontext. Dabei werden auch Gründe (Intentionen) für das Projekt genannt.

Projektgruppe II erstellte eine umfassende Dokumentation ihres Projekts mit MoLab. Diese ist wie ein How-to aufgebaut und enthält fünf Fotos. Sie zeigen den Sensor, die mit Krokodilklemmen verkabelte Hardware, die zusammengelötete Hardware mit aktiven Aktuatoren (z.B. Display), das Artefakt (mit T-Shirt als Korpus) am Körper in Aktion mit versteckter Hardware (siehe Abbildung 7.13), sowie das Artefakt am Körper mit offen gelegter Hardware. Die Beschreibung enthält vorwiegend technische Informationen wie eine Materialliste, eine Beschreibung der technischen Funktionalität und die Hardwarekonfiguration (Pinnnummern und Komponenten). Außerdem enthält das Projekt textuellen Arduino-Code.[13] Dieser Code besteht aus dem Programm, ergänzt durch Kommentare zur Funktionsweise. Die Hardwarekonfiguration wird in den Kommentaren auch erwähnt, die Anschlüsse sind aber auf den ersten Blick nicht ersichtlich. Dieses Projekt bildet daher mit Fotos vorwiegend die Hardwarekonfiguration und den Artefaktkorpus ab, also statische Merkmale. Da das T-Shirt auf Fotos am Körper getragen wird, ist auch sein Kontext ansatzweise abgebildet.

Abbildung 7.14 zeigt einen quantitativen Überblick der Ergebnisse.

7.3.4 Zusammenfassung und erste Schlussfolgerungen der Modellanalyse

Im Folgenden werden die Ergebnisse sortiert nach Methoden/Tools, Perspektiven und Modellarten zusammen gefasst. Die Diagramme in Abbildung 7.15 zeigen einen Überblick der quantitativen Auswertung. In der untersten Zeile wird die Verteilung sowohl auf die Modellkategorien Bildmodelle, textuelle Modelle und die spezielle Form des Amici-Programmcodes dargestellt. In den Reihen werden sie jeweils auf die einzelnen Tools bzw. Methoden bezogen. In der ersten Spalte sind die Verteilungen der Perspek-

[12] Das Kleidungsstück soll laut Beschreibung Männer darin hindern, Frauen zu nahe zu kommen.

[13] Warum die Teilnehmenden diesen bevorzugten, ist nicht bekannt. Es ist anzunehmen, dass sich ihre Projektidee nicht mit Amici umsetzen ließ.

tiven auf die jeweiligen Dokumentationseinheiten der einzelnen Modellierungswerkzeuge abgebildet.

Methoden und Tools

Auf Fotos und in den Notizen der vorgespielten Szenarien wurden recht unformal vorwiegend der Artefaktkorpus dargestellt bzw. beschrieben und teilweise dessen interaktives Verhalten angedeutet. Obwohl es sich um Projektideen handelte, wurden pragmatische Merkmale nicht ausdrücklich erwähnt.

Die untersuchten Skizzen bildeten im Wesentlichen den zu modellierenden Artefaktkorpus sowie Hardwareelemente (bzw. deren Schnittstellen) ab, was aber bei einem Teil der untersuchten Skizzen durch Vorlagen evoziert worden war. Teilweise wurden auch Interaktionsmöglichkeiten und in wenigen Fällen Intentionen dargestellt. Die Skizzen enthalten ikonische und schematische Elemente und können so in verschiedene Unterkategorien der Bildmodelle eingeordnet werden. Auch enthalten viele abundante, erklärende textuelle Modelle.

Mit Amici wird ein Modell formaler Attribute des Artefakts erstellt. Visuelle Programme als Mischmodelle aus symbolischen Darstellungsmodellen, ikonischen (auch schmematischen) Bildmodellen und textuellen programmiersprachlichen Modellen enthalten überraschenderweise nicht nur die Perspektiven Programm und Hardwarekonfiguration, sondern auch unformale und pragmatische Attribute, i.d.R. durch die Verwendung entsprechender Methodennamen.

Mit MoLab und dem VirtualLab erstellte Modelle können auch auf Artefaktebene betrachtet werden. So bildet das VirtualLab eine Projektdokumentation aus ikonischem Bildmodell und textuellen Modellen sowie Programmcode ab, die auch als ein Modell betrachtet werden kann und die aus verschiedenen einzelnen (oben beschriebenen) Modellen besteht. Als Ganzes bilden VirtualLab-Projekte die Perspektiven auf das Artefakt relativ ausgewogen ab. Allerdings (was aus der Analyse nicht hervorgeht) werden diese Modelle erst am Ende des Prozesses erstellt, sind also nur Modelle des fertigen Artefakts.

Mit MoLab können ähnliche Modelle wie mit dem VirtualLab erstellt bzw. dokumentiert werden. Darüber hinaus können mehrere zu einem Projekt entstandene MoLab-Einträge, die zusammen auf dem Modelboard abgebildet sind, den Modellierungsprozess abbilden. Generell enthalten die untersuchten MoLab-Einträge eine ähnliche Verteilung der Perspektiven wie die Projektdokumentationen im VirtualLab. Die Modelboards der einzelnen MoLab-Projekte enthalten nur in einem der drei untersuchten Fälle mehr als einen MoLab-Eintrag. In diesem Fall wurde dadurch der Entstehungsprozess abgebildet, wobei die einzelnen MoLab-Einträge immer den Programmcode, aber nicht immer Bilder und noch seltener Texte enthielten. Auffällig an den einzelnen Einträgen des Prozesses ist, dass jeweils immer nur das dokumentiert

wurde, was beim jeweiligen Termin geändert worden (meist der Programmcode) oder neu entstanden war (mehr dazu in Abschnitt 7.4.8).

Modellperspektiven

Modelle mit der Perspektive auf das interaktive Verhalten wurden verhältnismäßig wenig erstellt bzw. festgehalten. Das mag an den eher geringen Möglichkeiten, mit statischen Modellen und wenig Aufwand einen Interaktionsablauf festzuhalten, liegen. Interaktion wurde am ehesten in textuellen Modellen abgebildet. Auch in einigen der Bilder, die beim Vorspielen von Szenarien zur Ideenfindung entstanden, können Interaktionsattribute gefunden werden, was darauf hindeutet, dass diese Methode zu Modellen für interaktives Verhalten führen kann.

Attribute der Hardwarekonfiguration wurden hauptsächlich abgebildet in Amici-Code, aber auch in Bildmodellen in Form von Fotografien und Skizzen. In Letzterem wurden Hardwareelemente auch durch abundante textuelle Modelle zusätzlich beschrieben und damit besser erkennbar. Auf Fotografien wurden Hardwaremodelle, die den Zusammenbau während des Konstruktionsprozesses zeigen (vorwiegend in MoLab, weniger im VirtualLab), abgebildet.

Die Perspektive auf das dynamische formalisierte Verhalten des Artefakts wurde hauptsächlich abgebildet durch Programmcode, der in den untersuchten Fällen aufgrund der Rahmenbedingungen der Workshops fast ausschließlich mit der Visuellen Programmiersprache Amici erstellt wurde.

Selten fanden sich in beschreibenden Texten einzelne Phrasen, die Formalisierungen enthalten oder an Pseudo-Code erinnern, indem z.B. statt allgemeinsprachlicher Adjektive (z.B. „dunkel“) formalisierte numerische Sensorwerte genannt und in Bezug zu Aktionen gesetzt wurden.

Der Artefaktkorpus wurde überwiegend in Fotografien und Skizzen, also Bildmodellen mit ikonischem bzw. teilschematischem Charakter dokumentiert. Auch in textuellen Modellen, d.h. Beschreibungen oder Methodennamen im Amici-Code, fanden sich mitunter Hinweise auf die Form und das Aussehen des Artefakts.

Pragmatische Merkmale wurden vergleichsweise selten abgebildet. Am ehesten wurden Intentionen über textuelle Modelle dokumentiert. Modellsubjekte waren am ehesten auf Fotos zu finden. Der Kontext, dem das Artefakt entspringt und/oder für den es gemacht wurde, wurde gleichermaßen oft in Bildmodellen und textuellen Modellen abgebildet.

Modellarten

Die untersuchten Modelle lassen sich in verschiedene Unterkategorien der graphischen und der textuellen Modelle einordnen. Bildmodelle wurden vielfältig genutzt, um Artefaktkorpus, Hardwarekonfigurationen und Modellsubjekte durch Fotos festzuhalten.

Sie finden sich aber auch als Skizzen, kombiniert aus eher formalsprachlichen Symbolen mit unformalen ikonischen Abbildungen, welche vorwiegend den Artefaktkorpus und Hardwareelemente abbilden. Textuelle Modelle wurden seltener erstellt als Bildmodelle. Meist sind sie umgangssprachlich formuliert. Textuelle Modelle tauchen aber auch als Stichworte auf, die Skizzen mit abundanten textuellen Attributen versehen und so oft Attribute der Perspektiven Interaktion und Hardware darstellen.

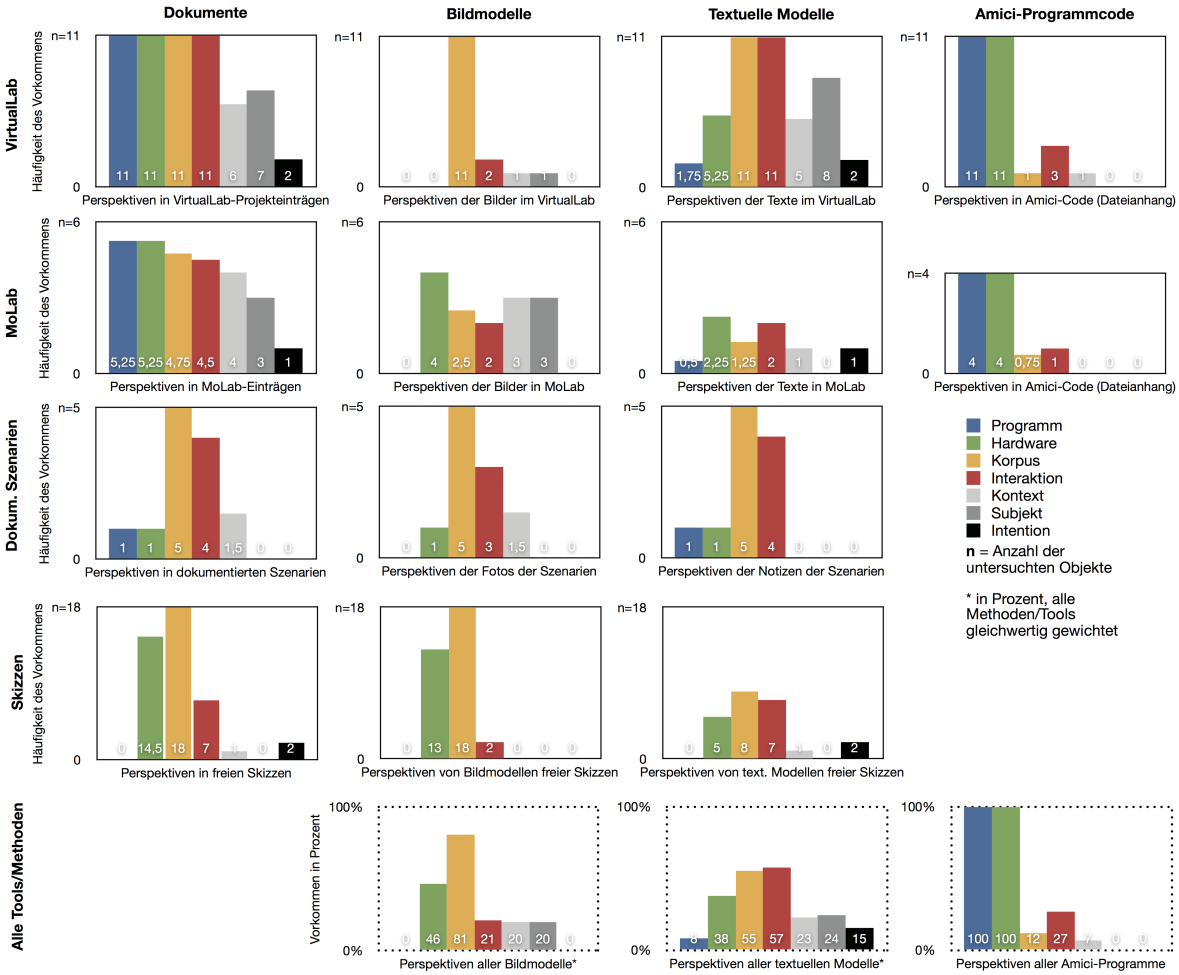


Abbildung 7.15: Quantitative Ergebnisse: Vorkommen von Perspektiven je Methode bzw. Tool und Modellart (vergrößerte Abbildung in Anhang B.1)

Einschränkung der Ergebnisse

Die Modellanalyse sagt vorwiegend etwas darüber aus, aus welchen Perspektiven und mit welcher Art von Modellen die Modellierenden mit den zur Verfügung stehenden Mitteln ihr stofflich-digitales Artefakt abbildeten. Die Ergebnisse müssen in Zusammenhang mit den verwendeten Methoden und Tools gesehen werden und den Funk-

tionen, die diese anbieten (und den Intentionen, die hinter deren Konzeption stecken). Auch waren die Teilnehmenden des Workshops „Robot DIY“ älter als die der übrigen Workshops und befanden sich schon im Studium. Die Analyseergebnisse können daher nur bedingt etwas darüber aussagen, ‚welche Modelle‘ junge Maker_innen im Allgemeinen erstellen. Eher legen sie dar, *wofür* und teilweise auch *wie* die zur Verfügung stehenden Mittel genutzt wurden. Sie zeigen Tendenzen auf, wie Amateur_innen formale Eigenschaften abbilden und welche Perspektiven sie bevorzugt dokumentieren. Die Modellanalyse kann auch nur bedingt Aussagen über den tatsächlichen Informationsgehalt der Modelle und ihre Vollständigkeit machen, wie diese von anderen verstanden werden und ob andere Personen mit den Modellen das Artefakt erfolgreich nach- oder weitermodellieren könnten.

Die Auswertung gibt einen Eindruck, wie die vorhandenen Tools und Methoden von jungen Amateur_innen zum Erstellen und Dokumentieren von Modellen beim Umsetzen ihrer stofflich-digitalen Artefakte genutzt werden und welche Modellarten bevorzugt werden. Offen bleiben die Beweggründe, die hinter den Modellen stecken, warum und wofür sie so und nicht anders erstellt wurden. Um diese quantitativen, größtenteils dekontextualisierten Ergebnisse besser einordnen zu können, aber auch um mehr zu den in Abschnitt 6.1.4 genannten Anforderungen an Modellierungswerkzeuge zu erfahren, werden die Erkenntnisse im nächsten Abschnitt um eine Fallstudie erweitert, in der die Benutzung von MoLab qualitativ ‚im Feld‘ untersucht und ausgewertet wird.

7.4 Modellierungstools in der Praxis

MoLab wurde als ein Modellierungstool konzipiert, das insbesondere die Anforderungen Flexibilität, Perspektivenvielfalt und -trennung, Modellnetze und Modellschnittstellen fokussiert, die bei den Tools Amici und VirtualLab weniger ausgebildet scheinen (siehe Abschnitt 6.5). MoLab orientiert sich lose am Prinzip der How-tos. Es bietet verschiedene Möglichkeiten graphische und textuelle Modelle sowie Dateianhänge einzubinden und diese einzelnen Modelle ähnlich einem Storyboard als den Prozess abbildende Modellnetze darzustellen. MoLab ist nicht nur ein Tool, das einzelne Modelle aufnimmt, sondern es kann auch Modelle des gesamten Modellierungsprozesses bzw. einzelner Schritte abbilden. Außerdem ist das Erstellen eines MoLab-Eintrags einem Projekteintrag im VirtualLab sehr ähnlich. Daher erlaubt eine Auswertung der Modellierungspraxis mit MoLab auch Einblicke in Amici und Rückschlüsse auf das VirtualLab.

MoLab wurde in drei Workshops erprobt, von denen der erste evaluiert wurde. Leitend waren dabei die Untersuchungsfragen, als wie geeignet sich die Implementierungen der Anforderungen sowie das kleinteilige, an How-tos und Storyboards angelehnte Prinzip für das Abbilden von Modellierungsprozessen in der Praxis erweisen.

Auch interessierte, welche Elemente oder Funktionen ein Modellierungstool enthalten sollte, so dass wesentliche Aspekte des Artefakts nachvollziehbar abgebildet werden können.

Die mit MoLab erstellten Modelle wurden schon in Abschnitt 7.3 analysiert hinsichtlich der Perspektiven, die sie zeigen. Mit MoLab erstellte Modelle (hauptsächlich MoLab-Einträge und einzelne Elemente wie Bilder, Text und Programmcode) und die Ergebnisse der Modellanalyse werden herangezogen, um die Beobachtungen und Interviewaussagen zu ergänzen.

7.4.1 Rahmenbedingungen

Inwiefern sich die in MoLab implementierten Funktionen für junge Amateur_innen beim Modellieren stofflich-digitaler Artefakte eignen, wurde in einer Fallstudie[14] untersucht. Dazu wurde über fünf Wochen hinweg ein Workshop als freiwilliges Angebot an einer Schule von der Autorin dieser Arbeit durchgeführt (im weiteren Verlauf ‚Durchführende' genannt). Die Termine dauerten jeweils 90 Minuten und fanden in ein- bis zweiwöchentlichem Abstand statt.[15] Es waren jeweils vier bis fünf Jugendliche im Alter von 15-16 Jahren anwesend, über den ganzen Zeitraum nahmen insgesamt vier Schüler und drei Schülerinnen teil. Vier der Teilnehmenden wollten eine Tasse (bzw. Becher) ‚intelligent' machen. Dieses Projekt wird hier als Fallstudie herangezogen. Zwei weitere Workshopteilnehmer, die nicht an diesem Projekt beteiligt waren und daher nicht in der Fallstudie berücksichtigt werden, waren unregelmäßig anwesend und bevorzugten es, mit wechselnden Sensoren und Aktuatoren zu experimentieren statt ein Projektziel zu verfolgen. Alle Teilnehmenden hatten schon mit Processing (siehe Abschnitt 2.1.2) programmiert. Ihnen war freigestellt, ihr Projekt mit der Programmiersprache Arduino (verwandt mit Processing) oder mit Amici zu programmieren. Die Gruppenmitglieder des Tassen-Projekts hatten noch keine Erfahrung im Umgang mit Arduino oder vergleichbaren Technologien und bevorzugten es, ihr Artefakt mit Amici zu programmieren.

Die Termine der Workshops bestanden im Wesentlichen aus freiem Arbeiten an den Projekten. Die Durchführende stand bei Fragen zur Verfügung, versuchte aber sonst möglichst wenig zu intervenieren oder anzuleiten. Die Teilnehmenden wurden gebeten, ihre laufenden Projekte in MoLab zu dokumentieren. Der erste Termin begann mit einer Vorstellung der Hardwareelemente und einer sehr kurzen Vorstellung von MoLab, ohne die Funktionen im Detail zu zeigen. Als Einstieg probierten alle aus, Sensoren, Schalter und Aktuatoren an ein Arduino- (bzw. LilyPad-)Board anzuschließen und mit einem einfachen Programm abzufragen bzw. anzusteuern. Beim zweiten Termin fand ein Brainstorming zum Thema „Objekte intelligent machen" anhand kon-

[14] Englisch: Case Study (siehe auch Lazar, Feng & Hochheiser, 2010).

[15] Im Gegensatz zu einem mehrtägigen durchgängigen Workshop entsprach diese Zeitrahmung vermutlich eher der Praxis vieler Maker_innen, die Projekten in ihrer Freizeit mit Unterbrechungen nachgehen.

kreter, von der Durchführenden mitgebrachter Objekte statt: T-Shirts, Schlüsselbänder, ein defekter Ventilator, ein Kaffeebecher. Inspiriert davon wurden Projektziele entwickelt. Die Fallstudiengruppe wollte daraufhin eine intelligente Tasse erstellen, die die richtige Trinktemperatur des Inhalts (nicht zu heiß oder zu kalt) anzeigt, um sich am Getränk nicht zu verbrennen, aber auch nicht die richtige Temperatur zu verpassen.

Die Durchführende brachte zu allen Workshopterminen Materialien und Laptops mit. An die Laptops waren bewegliche Webcams angeschlossen. Bastel- und Elektronikmaterialien, die zusätzlich gewünscht wurden, wurden auf Anfrage bereitgestellt. Die Hardwareteile (Mikrocontroller, Sensoren, Aktuatoren) mussten bei jedem Termin wieder neu zusammengesteckt werden, da die Durchführende die zusammengesteckten Projekte so nicht transportieren konnte. Dies diente aber auch als Anreiz, sie mit MoLab zu dokumentieren. Auch befanden sich in MoLab bereits Anleitungen für häufig benötigte Schritte (z.B. Schema zum Anschließen eines Sensors), die von der Durchführenden eingestellt worden waren (siehe Abbildung 6.11).

7.4.2 Datenerhebung

Die auszuwertenden Daten wurden mittels teilnehmender Beobachtung und einem Interview erhoben. Auch wurden die Dokumentationen, die mit Molab erstellt worden waren, herangezogen.

Teilnehmende Beobachtung mit Protokollen

Die Plattform MoLab wurde zwischen den einzelnen Erprobungsterminen aufgrund der Erkenntnisse aus den Beobachtungsprotokollen der vorangehenden Termine in Anlehnung an design-based research angepasst. Dazu wurden – wie bereits in Abschnitt 6.2.2 erwähnt – nach den Terminen die jeweils relevanten Beobachtungen in eine Liste aufgenommen, mögliche Änderungen erwogen und begründet. Bei den folgenden Terminen wurde durch Beobachtung verifiziert, ob die Änderungen den gewünschten Effekt hatten.

Die Methode der teilnehmenden Beobachtung wurde gewählt, weil sie eine große Nähe zum Forschungsgegenstand erlaubt, da an der Situation der Beobachteten teilgenommen wird. Da die teilnehmende Beobachtung insbesondere geeignet ist, wenn „der Gegenstand in soziale Situationen eingebettet ist“, „der Gegenstandsbereich von außen schwer einsehbar ist“ und „die Fragestellung eher explorativen Charakter hat“ (Mayring, 2002, S. 83), schien diese Methode für den vorliegenden Fall passend. Ein Nachteil der teilnehmenden Beobachtung ist jedoch, dass die Offenheit und Involviertheit Ergebnisse beeinflussen kann (Mayring, 2002).

Beobachtungsprotokolle teilnehmender Beobachtung dienen dazu, das Beobachtete, Gehörte und Erlebte festzuhalten. Da sie nachträglich erstellt werden, beruhen sie auf Erinnertem. Deshalb können sie nicht als wiedergabegetreu behandelt werden,

sondern als vom Beobachtenden und Protokollierenden sinnstiftend verdichtet und konstruiert (Lüders, 2008).

Die Durchführende des Workshops hielt ihre Beobachtungen in Beobachtungsprotokollen fest. Um die Beobachteten nicht zu verunsichern und da dies auch in der Situation als Tutorin nicht immer möglich war, wurden die Notizen für das Protokoll meist nach und nicht während der Veranstaltung erstellt. Die Beobachtungen wurden unmittelbar nach der Veranstaltung anhand eines Leitfadens stichpunktartig als zusammenfassendes Protokoll aufgeschrieben. Zusätzlich wurden weitere Gedanken und Vorkommnisse notiert. Darüber hinaus wurden anschließend erste Ideen für Verbesserungen gesondert notiert. Ansonsten dienten die Protokolle dazu, den Kontext der oben analysierten Modelle und des Interviews herzustellen und diese zu ergänzen. Sie standen aber nicht im Mittelpunkt der Auswertung. Zur Orientierung wurde ein offener Beobachtungsleitfaden erstellt, der Raum für neue, unerwartete Beobachtung ließ (siehe Anhang C).

Interview

Eine Woche nach dem letzten Termin wurde ein 20-minütiges Auswertungsgespräch geführt.[16] Die Teilnahme daran war freiwillig. Ziel war es, mehr über die Hintergründe der Modelle und die Praxistauglichkeit der in MoLab implementierten Modellierungsoptionen zu erfahren als die erstellten Modelle und die Protokolle verraten können. Als Diskussionsgrundlage lagen auf Papier ausgedruckte Screenshots von MoLab-Seiten vor. Diese zeigten die Projekt-Übersichtsseite, Modelboards zweier Projekte (darunter das der interviewten Gruppe), dazugehörige MoLab-Einträge und die Eingabemasken für MoLab-Einträge. Das Gespräch wurde aufgenommen und anschließend transkribiert. Diese Daten wurden ausgewertet, um den in Abschnitt 7.1 aufgeworfenen Fragen nachzugehen.

Das Auswertungsgespräch kann als halbstrukturiertes und offen problemzentriertes Interview (Mayring, 2002) bezeichnet werden. Demzufolge orientierte es sich an einem Leitfaden, anhand dessen die Fragen gestellt wurden (siehe Anhang D). Bei der Formulierung und hinsichtlich Nachfragen, Kommentaren und ähnlichem wurden Fragen nicht standardisiert behandelt, sondern der Situation angepasst. Dies hat nach Mayring (2002) den Vorteil, dass die Befragten ihre Gedanken besser entwickeln können. Da das Interview in der Gruppe stattfand, enthält es auch Merkmale einer Gruppendiskussion. In Gruppendiskussionen sind Meinungen und Einstellungen u.U. besser erhebbar als Sinnstrukturen, die einer sozialen Situation entspringen (Mayring, 2002). Da die mit MoLab erstellten Modelle die Diskussionsgrundlage des Auswertungsgesprächs bildeten, erschien eine standardisierte Interviewform weniger geeignet. Denn es war zu erwarten, dass neue, nicht vorhersehbare Aspekte im Ge-

[16] Dabei waren nur drei der vier Teilnehmer_innen der Projektgruppe anwesend.

spräch aufkommen. Als Gruppe konnten die Interviewten sich in ihren Erinnerungen ergänzen, widersprechen oder gemeinsam Gedanken weiterentwickeln.

7.4.3 Vorgehen

Wie in Abschnitt 7.2.2 bereits ausgeführt, wurde zur Auswertung des Modellierens mit MoLab eine inhaltlich-strukturierende Analyse in Anlehnung an Mayring (2010) gewählt. Die Reihenfolge der Analyseschritte entsprach dem in Abbildung 7.1 (S. 139) im rechten Zweig dargelegten Ablauf. Aufgrund der besseren Lesbarkeit und des Verständnisses wird das Vorgehen hier in leicht geänderter Reihenfolge beschrieben.

Die Auswertung des Interviews und der Protokolle orientierte sich an Ausdrucken auf Papier der in MoLab erstellten Modelle und an den im Interview genannten Beweggründen für das Erstellen der Modelle. Ziel war es, Modellierungsprozesse beim Making und die Eignung von MoLab zu ergründen. Dabei war die Analyse bewusst offen für neue Beobachtungen. Um die Modellierungsprozesse und die darin vorkommenden Beweggründe für die Entstehung der Modelle zu erfassen, schien ein Analyseverfahren, das sich an quantitativen Maßstäben orientiert, weniger geeignet, wie in Abschnitt 7.2 bereits dargelegt wurde.

Als Material dienten das Auswertungsgespräch als transkribierter Text sowie die in MoLab erstellten Webseiten mit Texten, Bildern und Dateianhängen. Außerdem wurden die Beobachtungsprotokolle (Texte) und die Ergebnisse der Modellanalyse (siehe Abschnitt 7.3) als Kontextmaterial herangezogen. Als Analyseeinheiten wurden festgelegt: Bilder (erkennbare Details bis Gesamtbild), Text (Präposition bis Gesamttext als Einheit), Programmcode (Einzelblock und dessen Text oder Bild bis hin zum Programm als Ganzem), Webseiten (einzelne Elemente darin und die Seite als ganzes Element).

Zur Untersuchung der Bedienbarkeit – oder Usability – von MoLab stellte sich die Frage, wo bei der Benutzung von MoLab Probleme auftraten, welche Funktionen und Interface-Elemente nicht verstanden wurden und was als brauchbar empfunden wurde.

Neben den in Abschnitt 7.1 genannten Evaluationsfragen orientierte sich die Analyse an den einzelnen Anforderungen und Annahmen, auf denen die Konzeption von MoLab beruht (siehe Abschnitt 6.1.4 und 6.5.2).

Als Analysekategorien wurden zunächst die Dimensionen „Bedienbarkeit“ und „Modellierung“ unterschieden. Die Textpassagen des transkribierten Interviews und der Protokolle, die der Bedienbarkeit zugeordnete waren, wurden dann einzelnen Webseiten von MoLab als Unterkategorien zugeordnet. Aussagen zu den Modellen und dem Modellierungsprozess wurden in diese Unterkategorien eingeteilt: „Intentionen und Kontext“, „Bedeutung von Fotos und Skizzen“, „Kollaborationsmöglichkeiten“, „Modelboard-Struktur“, „Modelboard-Kategorien“, „Bedeutung von Texten und Annotationen“. Die Interviewaussagen wurden, soweit möglich und für ein tieferes

Verständnis nötig, mit den Beobachtungsprotokollen als Kontextinformation in Bezug gesetzt. Außerdem wurden den Interviewpassagen die jeweils diskutierten Elemente und Funktionen von MoLab (bzw. vorliegende Screenshots) zugeordnet und bei der Interpretation als Gliederung herangezogen.

Im Folgenden werden die Ergebnisse zusammengefasst. Diese werden durch Zitate aus dem Interview ergänzt, die zur besseren Lesbarkeit mit Satzzeichen versehen und ohne Transkriptionssymbole dargestellt werden.[17] Die Namen der befragten Workshopteilnehmer_innen wurden als T1, T2 und T3 anonymisiert. Zunächst werden die Ergebnisse der Usability-Auswertung vorgestellt.

7.4.4 Auswertung der Bedienbarkeit

Die Bedienbarkeit wurde u.a. ausgewertet, um Aufschluss darüber zu bekommen, ob bestimmte Funktionen ggf. nicht benutzt wurden, da sie nicht handhabbar waren oder nicht erkannt wurden. Dazu wurde gegen Ende des Auswertungsgesprächs die Bedienbarkeit von MoLab thematisiert. Um die Fokussierung auf Aspekte der Bedienbarkeit zu unterstützen, wurde eine „Dot Voting"-Technik eingesetzt. Dies wurde bereits in einem Forschungsprojekt, an dem die Autorin beteiligt war, für Fokusgruppen mit Teenagern für die oben beschriebenen Zwecke entwickelt und erprobt (Katterfeldt, Zeising & Schelhowe, 2012). Diese Methode knüpft an die bei der Zielgruppe beliebte Praktik des ‚Likens' (deutsch ‚gefällt mir') in Social Networks an. Dazu wurden die Teilnehmenden gebeten, mit verschiedenfarbigen Klebepunkten, die ‚Likes' und ‚Dislikes' repräsentierten, Elemente oder Funktionen auf den vorliegenden Screenshot zu bewerten. Anschließend wurden die Screenshots gemeinsam betrachtet und die Teilnehmenden erläuterten ihre Likes und Dislikes, wobei alle ihre Meinung einfließen lassen konnten und die Forscherin die Gelegenheit hatte, Rückfragen zu stellen. Durch diese Methode werden daher nicht nur quantitative Daten erhoben, sondern es findet eine Rückkopplung mit qualitativer Aussage statt. Darüber hinaus werden die Teilnehmenden dazu angeregt, eigene Verbesserungsvorschläge zu beschreiben. Die Methode hat auch den Vorteil, dass jede_r die eigene Meinung in Form der Klebepunkte zunächst diskret und ohne große Hürde kundtun kann, anschließend aber auch aufgefordert wird, die Entscheidung zu begründen, so dass auch eher zurückhaltende Teilnehmer_innen zu Wort kommen.

In diesem Teil des Auswertungsgesprächs bestätigten sich im Wesentlichen die protokollierten Beobachtungen hinsichtlich der Benutzbarkeit. Die Startseite, das Modelboard und MoLab-Einträge wurden als übersichtlich und informativ bewertet, u.a. wegen der sichtbaren Bilder. Teilweise traten während des Workshops Probleme mit unbeabsichtigtem Verschieben der MoLab-Einträge auf dem Modelboard auf. Die Befragten sprachen sich deshalb für ein freies Bewegen der MoLab-Einträge auf dem Modelboard

[17] Kurze unverständliche Phrasen und Pausen wurde ausgelassen. Ergänzungen oder Kommentare, um den Zusammenhang herzustellen, stehen in [ ].

aus, schlugen aber ein eindeutiges Raster wie in Amici vor (ohne Zwischenversetzung). Positiv wurde bewertet, dass Fotos auf verschiedene Weise integriert werden können. Die Funktion zur Annotation von Fotos wurde nicht erkannt. Bei der Benutzung des Modelboards zur Erstellung von Modelldokumentationen bzw. Artefakt-Modellen und der nicht erfolgten Annotation von Bildern spielte also auch die unzureichende Bedienbarkeit eine Rolle.

7.4.5 Auswertung des Modellierens mit MoLab

Im Folgenden wird die Auswertung der Modellierungsprozesse mit MoLab beschrieben. Dabei werden auch Ergebnisse der inhaltlichen Modellanalyse (siehe Kapitel 7.3) herangezogen.

Perspektivtrennung und Kategorien

Das so genannte Modelboard in MoLab wurde ursprünglich so konzipiert, dass MoLab-Einträge den Kategorien Artefaktkorpus, interaktives Verhalten, Programmierung oder Hardwareaufbau zugeordnet werden können.

Die MoLab-Einträge der Fallstudiengruppe (und auch die sonstigen analysierten) waren nicht kategorisiert. Sinn und Zweck der Einteilung in Kategorien wurden erst im Interview erkannt. Diese Einteilung scheint also nicht intuitiv zu sein. Für das eigene, als klein bezeichnete Projekt sahen die Befragten eine solche Einteilung als nicht nötig an, befürworteten sie aber für umfangreichere Projekte:

> T2: „(...) allgemein ist ja unseres ein bisschen kleineres Projekt. Ich glaub wenn man jetzt wirklich als Ziel hat so ne fertige Tasse, so ne vermeindbar verkaufbare Tasse zu machen, ich glaub, dann ist das praktischer, wenn man auch unterschiedliche Gruppen hat, die sich dann, eine um das Visuelle kümmern und die andere um die Software, (...) und dann laden halt manche unter Kategorie Software [gemeint ist ‚Programm'] was hoch und manche unter Kategorie Hardware."

Auf dem Modelboard wurden Einträge nicht gezielt angeordnet. Dies liegt möglicherweise auch daran, dass die Bedienbarkeit nicht optimal war, um die Funktion zu erkennen, zumal sich die Anordnungsmöglichkeit (freie Anordnung statt Spalten) im Laufe des Workshops änderte (siehe Abschnitt 7.4.6). Beobachtet wurde, dass während der Benutzung von MoLab die meiste Zeit eher ein MoLab-Eintrag geöffnet war. Das Modelboard diente zur Navigation durch fremde Projekte, um sich dort z.B. abzuschauen, wie eine RGB-LED angeschlossen und programmiert wird.[18]

Bildmodelle und textuelle Modelle

Wie die Modellanalyse zeigte, wurden mit MoLab vor allem Bildmodelle erstellt. Auch die Teilnehmenden des Tassenprojekts dokumentierten den Zustand ihres Artefakts

[18] Auf der Plattform befanden sich neben Anleitungen der Tutorin im Laufe des Workshops auch Projektdokumentationen der anderen Teilnehmenden.

mit vielen Fotos. Diese zeigten sie selbst, die mit Krokodilklemmen zusammengefügte Hardware, eine Skizze und den fertigen Prototyp. Mit der Skizze modellierten sie ihre Vorstellung von der Umsetzung ihres Projekts, hauptsächlich hinsichtlich dessen Korpusgestalt und der Hardwarekonfiguration. Die Skizze wurde in MoLab eingebunden, um die Idee zu veranschaulichen:

> T1: „Also wir haben einmal hier fotografiert eine Skizze von der äh Tasse, wie sie hätte aussehen können und wie wir uns das vorgestellt haben, einfach nur, um das noch mal zu veranschaulichen, was wir uns dabei gedacht haben."

Die Skizze dient als Vorbild, um die Soll-Vorstellung des Projekts abzubilden und ist somit ein Modell der Projektidee.

Ein Foto zeigt ein LilyPad, an das mit Krokodilklemmen ein Temperatursensor, eine RGB-LED und ein Piezo angeschlossen sind (ähnlich wie Abbildung 7.12). Im Interview wurde das Bild der Hardwarekonfiguration als nicht so aufschlussreich wie die Skizze bewertet, da es „Kabelsalat" zeige. Die Befragten gaben an, das Bild des Hardwareaufbaus ursprünglich zu dem Zweck erstellt zu haben, um das Projekt beim nächsten Termin in derselben Konfiguration wieder aufbauen zu können: „dass wir ungefähr sehen, wie es denn aussah, das was wir hatten", so T1. Sie stellten im Interview fest, dass das Foto dazu von zu schlechter Qualität sei und erinnerten sich, dass sie auch ohne das Bild in der Lage waren, ihre Hardwarekonfiguration wieder aufzubauen. Sie bemerkten daraufhin, dass das in MoLab hochgeladene Amici-Programm genug Informationen enthielt:

> T1: „Diese Kabelsalatbilder hier sind natürlich nicht so [bezogen auf die Skizze, s.o.], äh, aufschlussreich."
> T3: „Die haben wir eigentlich nur, damit wir es wieder aufbauen können oder?"
> T1: „Ja, aber im Endeffekt ging es ..."
> T3: [unterbricht] „So direkt so."
> T1: „ ... eigentlich auch so." [Zustimmung von T2 und T3]
> T1: „Ja ich glaub es hätte sogar, ja mh, eigentlich hat auch das Programm gereicht, wenn man gesehen hat, was wo angeschlossen war."

Die Hardwarekonfiguration wurde ursprünglich fotografiert, um das Projekt beim nächsten Termin in derselben Konfiguration wieder aufbauen zu können. Das Bildmodell bildet also den aktuellen Stand ab, zeigt ein informelles, unformales Abbild der Hardwarekonfiguration und soll als Vorbild für das Weitermodellieren beim nächsten Termin dienen. Die Abbildung kann also als ikonisches, graphisches Bildmodell eingestuft werden, enthält aber auch Informationen durch die Kabelverbindungen, die Nähe zu einem symbolischen Darstellungsmodell wie z.B. einem Schaltkreis zulassen, bzw. war es als solches gedacht. Das Foto wird aber als nicht so aufschlussreich wie die reduziertere Skizze angesehen. Mit Ausnahme der Skizze enthielten die Fotos keine textuellen Untermodelle wie z.B. Annotationen (abundante textuelle Modelle), sondern nur ikonisch dargestellte Attribute. Da es sich bei den Bildern der Hardware-Konfiguration um die noch vorläufige, noch nicht in das Produkt (eine Tasse) eingebaute Konfiguration handelt, sind strukturelle Merkmale (Kabelanschlüsse, Einzelteile

der Hardware) noch erkennbar. Details wie Pinnnummern können aus dem Foto aber nicht abgelesen werden – was letztendlich auch nicht notwendig war, weil diese Details im Amici-Programm dokumentiert waren, wie festgestellt wurde.

Die direkte Einbindung von Bildern des Modellierungsprozesses über eine (bewegliche) Webcam erwies sich als geeignet und wurde für alle eingebundenen Fotos genutzt. Von den anderen Funktionen zum Einbinden von Bildern (neue hochladen, bereits hochgeladene Bilder erneut einbinden, vom System bereitgestellte Hardwarebilder einfügen) wurde hingegen kein Gebrauch gemacht. Unter Heranziehen der Modellanalyse zeigt sich, dass die Webcam-Funktion dazu genutzt wurde, Fotos mit Modellattributen, die nicht in graphischen Abbildungen von Amici-Programmcode enthalten waren, abzubilden, wie z.B. Artefaktansichten und ansatzweise das Artefaktverhalten. Mit der Skizze wurde auch ein externes Bildmodell eingebunden. Auch wurde über Fotos der Kontext und wer die Modellsubjekte sind als pragmatische Modellmerkmale mitgeteilt.

Die Möglichkeiten von MoLab, textuelle Modelle zu erstellen, wurden wenig genutzt. In einem MoLab-Eintrag wurde eine kurze Beschreibung in das Beschreibungs-Textfeld eingetragen. Auch die Kommentarfunktion wurde nicht benötigt. Textuelle Modelle finden sich an anderen Stellen: in den Dateinamen der Programme, den Methodennamen, den Beschriftungen der Skizze, dem Titel (sofern eingetragen). Die Methodennamen im angehängten Amici-Programm und die Anmerkungen auf der Skizze können als abundante, erklärende textuelle Modelle eingestuft werden.

Textuelle Modelle spielten also beim Dokumentieren aktueller Projektzustände eine untergeordnete Rolle. Allerdings verweisen die Interviewten darauf, dass sie textuelle Beschreibungen sinnvoll fänden, wenn sie ihre Projekte mit anderen außerhalb des Workshops teilen würden (siehe unten).

Modelle erstellen für andere: Pragmatische Merkmale

Im Interview wurden die Teilnehmenden gefragt, ob oder inwiefern sie bei der Dokumentation anders vorgehen würden, wenn sie ein vergleichbares Projekt alleine umsetzen würden und dabei mit anderen über MoLab kommunizieren könnten.

> T1: „Vielleicht hätte man die Bilder ein bisschen übersichtlicher gestalten können, also dass man auch wirklich erkennt, was wirklich gemacht wurde (...).“
> T2: „Also, dass einer auch was daraus lesen kann. Vielleicht auch ein Foto machen und dann mit (...) Paint, Gimp oder was auch immer da halt Striche und Beschriftung, dass man das erkennen kann, was man da überhaupt abgelichtet hat.“

Mit dem in der Interviewfrage genannten Beispiel ändern sich die pragmatischen Merkmale der dokumentierten Modelle. Das *für wen* und *zu welchem Zweck* wird ein anderes. Die Teilnehmenden gaben an, dass sie, dem geänderten Zweck entsprechend, mehr Wert darauf gelegt hätten, dass wesentliche Modellmerkmale auf Bildern er-

kennbar sind. Zur Unterstützung schlagen sie vor, Fotos mit einem Bildbearbeitungsprogramm zu annotieren.

Auf die Frage, welche anderen Elemente sie ggf. noch benutzt hätten, befürworten sie, Text zu ergänzen:

> T1: „Ich glaube wenn man das mit verschiedenen Leuten auf der ganzen Welt macht, dann wärs sinnvoll da noch Text hinzuschreiben, einfach nur ein Bild oder ein Programm erklärt ja nicht immer gleich alles, da kann man auch noch dazu schreiben was man sich dabei gedacht hat, wie es im Endeffekt funktionieren soll."
> T2: „Oder ob einem irgendwas aufgefallen ist während dessen mans gemacht hat, irgendwas halt nicht funktioniert, dann kann man's hochladen und sagen was nicht funktioniert hat."

Mit textuellen Modellen sollen Intentionen, also pragmatische Merkmale, angegeben werden, um Bilder und Programme zu erläutern, aber auch, um den Zielzustand und wie das Artefakt funktionieren soll, zu beschreiben. Außerdem wird zusätzlicher Text im skizzierten Fall als sinnvoll erachtet, um Fragen zu stellen oder Auffälligkeiten anzugeben.

Um pragmatische Merkmale mitzuteilen, scheinen also kurze textuelle Modelle in Form von Mitteilungen und Annotationen am zweckmäßigsten zu sein.

7.4.6 Anpassungen von MoLab

Bereits während des ersten Workshops wurden – gemäß des an design-based-research angelehnten Vorgehens – Änderungen an MoLab vorgenommen. So wurden aufgrund des Beobachtungsprotokolls Probleme der Bedienbarkeit von MoLab behoben. Die Bedienbarkeit der MoLab-Einträge auf dem Modelboard wurde verbessert. Die Elemente erhielten ein ‚drag handle' zum einfacheren Verschieben; Bilder wurden direkt mit den Einträgen verlinkt. Da die Kategoriezuordnung nicht genutzt wurde, wurde das Spaltenprinzip bereits zum Ende des ersten Workshops abgeschafft zugunsten freier Anordnung (siehe Abbildung 7.16). Auch wurden Bezeichnungen beim Erstellen der MoLab-Einträge geändert. Da grundsätzlich nur Programme als Dateianhänge hochgeladen wurden, wurde z.B. diese Funktion in „Programmcode-Anhänge" umbenannt. So wurde auch der Tatsache Rechnung getragen, dass ein einziger MoLab-Eintrag das Projekt allumfassend – also zugleich aus mehreren der Perspektiven anstatt nur aus einer – dokumentieren kann.

7.4.7 Weitere Erprobungen von MoLab

Die überarbeitete Version von MoLab wurde später in zwei parallelen mehrtägigen TechKreativ-Workshops eingesetzt. Die dort erstellten Modelle wurden bereits in Abschnitt 7.3.3 analysiert. Die Autorin dieser Arbeit war bei diesen Workshops nicht anwesend und es fanden auch keine Auswertungsgespräche über MoLab statt. Deshalb

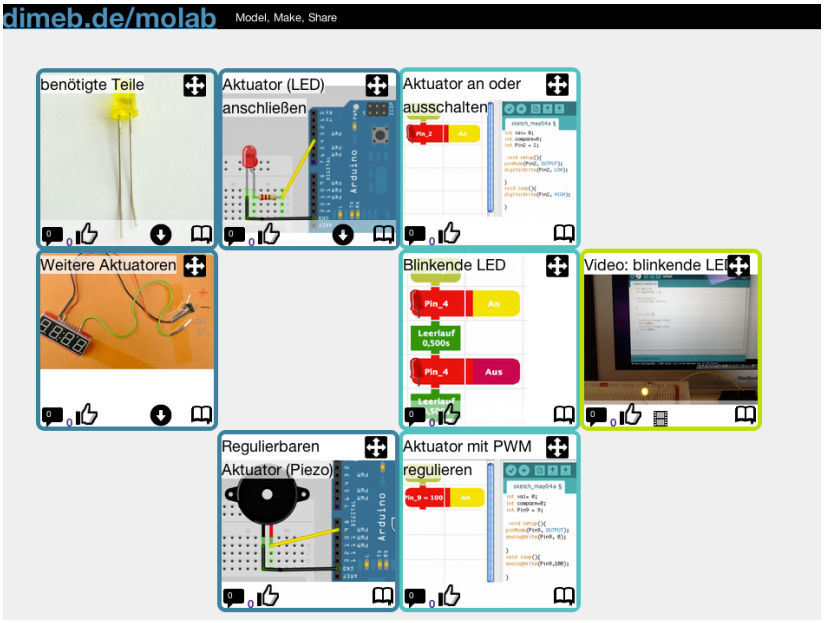


Abbildung 7.16: Überarbeitetes Modelboard in MoLab Version 2.0

sollen hier nur Eindrücke geschildert, aber keine systematische Auswertung vorgenommen werden. Sie basieren auf einem informellen Gespräch mit einer der Tutorinnen und auf den Eintragungen in MoLab.

Das Tool MoLab wurde von den Tutorinnen zu Beginn nicht explizit vorgestellt, die Teilnehmenden bekamen in erster Linie die Internetadresse der Plattform genannt und wurden individuell darauf hingewiesen, dass sie dort ihre Projekte dokumentieren können. Wie oben bereits erwähnt, nutzte eine Gruppe MoLab, um ihre Projektidee zu beschreiben. Die andere Gruppe erstellte eine How-to-artige Dokumentation ihres Projekts. Es liegt nahe, kann aber nur vermutet werden, dass diese Gruppe bereits Erfahrungen mit How-tos hatte. Wie in einem How-to finden sich, anders als mit dem Modelboard in MoLab intendiert, alle einzelnen Schritte in einem MoLab-Eintrag. Diese Gruppe benutzte nicht die Visuelle Programmierumgebung Amici, sondern lud textuellen Arduino-Code zu ihrem Projekt hoch. Dafür beschreiben sie im Beschreibungs-Textfeld die Pinnanschlüsse und Hardwarekonfiguration. Daraus kann gefolgert werden, dass sie den Arduino-Code (auch wenn dieser Kommentare enthielt) nicht für aufschlussreich genug hielten und deshalb die Hardwarekonfiguration zusätzlich beschrieben.

In diesen beiden Workshops benutzten insgesamt nur zwei Projektgruppen MoLab. Davon dokumentierte eine Gruppe relativ wenig. Dies bestätigt die Ergebnisse der Fallstudie, als dass externes Erstellen von Modelldokumentationen zu aufwändig scheint, unbeliebt ist und/oder sich schlecht in die Praxis integrieren lässt.[19]

[19] Allerdings bestand in den beiden späteren Workshops weniger Notwendigkeit MoLab zu benutzen, da diese an einem Stück stattfanden und keine mehrtägigen Pausen hatten.

Als ein Nebenergebnis sei noch bemerkt, dass für alle Workshops von den Tutor_innen Materialien und Beispiele für Hardwareanschlüsse und Programmierung[20] in MoLab eingestellt worden waren. Diese wurden während der Workshops oft als Ressourcen genutzt.[21] Insbesondere Bilder von Amici-Programmen wurden oft angesehen, obwohl z.B. auch Bilder von Hardwareaufbau und Videos der Funktionsweise vorlagen. Damit scheint MoLab auch den üblichen Nutzen einer How-to-Plattform erfüllen zu können.

7.4.8 Erkenntnisse über den Modellierungsprozess

Making wurde in Abschnitt 6.1.2 als iterativer, kreativer Prozess beschrieben. Die Fallstudie bestätigte diese Einschätzung im Wesentlichen. Am Anfang der Workshops entwickelten die Teilnehmenden eine Projektidee, die sie im Laufe des Prozesses als Skizze festhielten. Deren genaue Ausgestaltung änderte sich während der Umsetzung. Zum einen wegen technischer Einschränkungen (zu wenige so genannte PWM-Pinns am Board), zum anderen aber auch, weil sie neue Möglichkeiten entdeckten (z.B. Erzeugen von Mischfarben mit der RGB-LED).

Die einzelnen Artefakt-Modelle (die jeweils aus einem MoLab-Eintrag bestanden) zeigen ein auf wenige Attribute und hauptsächlich auf die Perspektiven Hardware und Programm reduziertes Modell (siehe Modellanalyse in Abschnitt 7.3.3). Verglichen mit den Möglichkeiten, die MoLab bietet, und mit dem Projekt als Ganzem erscheinen diese Modelle einseitig und sehr reduziert. Betrachtet man, womit die Modellierenden bei den einzelnen Terminen jeweils beschäftigt waren, so bilden die Artefakt-Modelle den jeweiligen Modellbildungsprozess zum jeweiligen Zeitpunkt ab bzw. das, was an Modellen geändert wurde und im Fokus des Making stand. Die Artefakt-Modelle bilden also die Änderungen ab, indem sie nur diejenigen Elemente enthalten, die aktuell verändert wurden. Der zweite erstellte MoLab-Eintrag, zu dem die Skizze mit der Projektidee gehört, enthält grundlegende Information zum Korpus der Tasse. Würde jemand anhand dieses MoLab-Eintrags das Projekt aufnehmen, wäre also die grundlegende Gestalt bekannt. Das Äußere des Artefakts entwickelte sich erst im Laufe des Prozesses durch verfügbares Material und weitere Abwägungen. Mit dem Projektziel verbundene Intentionen, Rahmenbedingungen und auftauchende Probleme werden mit den zur Verfügung stehenden Mitteln nicht beschrieben.

Beim Modellieren von eher kleinen Makerprojekten scheinen Entwicklung der Projektidee, Entwurf des Artefakts und dessen formalisierende (und algorithmisierende) Implementierung und materielle Umsetzung kaum getrennt voneinander stattzufinden. Es ist vielmehr ein fortlaufendes, iterierendes „reflection-in-action“ (Schön, 1983, S. 49), indem Prototypen entstehen und überarbeitet werden. Making, so wie es auch

[20] Diese enthielten nicht nur Code, sondern auch Screenshots von Amici-Programmen.

[21] Wie im privaten Kontext konnten die angehenden Maker_innen sich so auch an externen Projektdokumentationen orientieren.

in den TechKreativ-Workshops praktiziert wird, scheint vor allem ein kreativer Prozess zu sein. Durch das ständige Hinzulernen ändert sich der Erfahrungshorizont. Es wird deutlich, was möglich ist, aber auch, was nicht umsetzbar ist. In diesem Prozess ändert sich auch das Vorbild laufend. So hatte auch das Projekt „Tasse“, das anfangs von seinen Macher_innen „ProjektPeep“ genannt worden war, am Ende gar keine Piep-Funktionalität mehr. Durch das flexible Ziel gibt es keinen Grund, an einem bestimmten Punkt ein Vorbild zu manifestieren. Möglicherweise liegt der Reiz des Making gerade in diesem von kreativen Einfällen geleiteten Prozess. Ein planerisches Vorgehen mit einem festen Vorbildmodell scheint dem eher zu widersprechen.

7.4.9 Erste Schlussfolgerungen zum Nutzen von MoLab

Mit dem Modellierungstool MoLab können Modelle, die beim Selbermachen eines stofflich-digitalen Artefakts entstehen, dokumentiert und organisiert werden. Es wurde konzipiert und implementiert, um die Anforderungen Flexibilität, Perspektivenvielfalt und -trennung, Modellnetze und Modellschnittstellen zu erfüllen (siehe Abschnitt 6.1.4). Auf die generelle Eignung von MoLab und die Erfüllung der Anforderungen wird hier angesichts der Ergebnisse der Erprobung eingegangen.

MoLab vermochte das Erstellen von Bildmodellen anzuregen. Diese beinhalteten hauptsächlich Bildmodelle und hochgeladene Dateien des Programmcodes, der in jedem Fall erstellt werden musste. Für das Erstellen von Bildmodellen erwies sich die Anbindung an eine Webcam als förderlich. Sie wurde genutzt, um eine Reihe von Artefaktaspekten zu dokumentieren, darunter den Hardwareaufbau. Eine solche Kamera ist ein flexibles Instrument, um (extern erstellte) Modelle zu dokumentieren und gut in den Prozess des Making integrierbar.

Modellierungsfunktionen wie das Erstellen von Texten oder das Anordnen der MoLab-Einträge, wurden kaum oder gar nicht genutzt. Da dies die Benutzung eines externen Tools in einem Webbrowser erfordert, was sonst im Prozess nicht notwendig ist, scheint diese Art des Modellierens und Dokumentierens durch seine geringe Praxisintegration weniger geeignet zu sein. Auch die Trennung in Perspektiven erwies sich als wenig praktikabel. Für gewöhnliche, kleinere Projekte – so wie im vorliegenden Fall – scheint solch eine Perspektivtrennung also nicht anwendbar zu sein und widerspricht eher dem kreativen Prozess des Making.

MoLab-Einträge enthielten mehrere Modelle verschiedener Perspektiven gleichzeitig – z.B. als Amici-Programmcode und teilweise auch Fotos der Hardware und einer Skizze (siehe auch Modellanalyse) – und wurden nicht getrennt nach Perspektiven erstellt. So wurden innerhalb der MoLab-Einträge auch Schnittstellen zwischen den Modellen einzelner Perspektiven abgebildet.

Insgesamt wurde MoLab nicht so benutzt, wie es konzipiert worden war. Konstruieren und Dokumentieren, also Modellieren und das Abbilden dieser Modelle in Dokumenten, scheinen nicht gut zueinander zu passen. Ein zusätzliches Tool zum

Festhalten von Modellen im Modellierungsprozess, das sich nicht auf natürliche Weise in den Prozess integriert, scheint also in der Praxis nicht zielführend zu sein. Allerdings deckte die Evaluation Potenziale der Visuellen Programmierumgebung Amici als Modellierungs- und Dokumentationstool auf, die so nicht erwartet worden waren. Die Teilnehmenden konnten den visuellen Programmen genug Informationen entnehmen, um ihr Projekt wieder aufzubauen und fortzuführen. Dass Amici-Programme als Modelle mehr als eine Perspektive auf die Programmierung abbilden, hatte sich schon in der Modellanalyse (siehe Abschnitt 7.3) angedeutet. Anhand der Erfahrungen kann geschlussfolgert werden, dass Einfachheit und Praxisintegration, wie sie Amici mit sich bringt, wichtiger sind als Flexibilität im Sinne vieler verschiedener Funktionen, so wie von MoLab angeboten.

In Abschnitt 7.5.5 werden die Anforderungen anhand der Gesamtergebnisse aus Modellanalyse und Erprobung revidiert.

7.5 Zusammenfassung der Evaluationsergebnisse und Konsequenzen

Mit der Evaluation sollte untersucht werden, welche Perspektiven und AMT-Kategorien die dokumentierten Modelle aufzeigen, wie formale Eigenschaften abgebildet werden, wie sich die von MoLab implementierten Anforderungen in der Praxis bewähren und welche Elemente oder Funktionen ein Modellierungstool enthalten sollte, so dass wesentliche Aspekte des Artefakts nachvollziehbar abgebildet werden können (siehe Abschnitt 7.1). In diesem Abschnitt werden die Ergebnisse beider Untersuchungen auf diese Fragen bezogen, um Erkenntnisse für die generellen Forschungsfragen dieser Arbeit ableiten zu können. Aufgrund der teilweise neuen und unerwarteten Ergebnisse wird außerdem in einem eigenen Abschnitt auf das Potenzial Visueller Programmierumgebungen als Modellierungstools eingegangen.

7.5.1 Modelltheoretische Einordnung der untersuchten Modelle

Die erste Evaluationsfrage lautete: Was kennzeichnet die in TechKreativ-Workshops dokumentierten Modelle, welche Perspektiven enthalten diese und wie sind die Modelle nach AMT einzuordnen? Zunächst werden die Modellkategorien, in die die untersuchten Modellen eingeordnet werden können, zusammengefasst. Anschließend werden die Perspektiven, die die Modelle einnehmen, auf diese Kategorien bezogen.

Einordnung nach AMT

Die untersuchten Modelle waren vorwiegend graphische und textuelle Modelle bzw. waren zusammengesetzt aus verschiedenen Elementen dieser Modellarten. Sie wer-

den genutzt, um ein technisches Modell abzubilden – nämlich das gerade entstehende stofflich-digitale Artefakt.[22]

Unter den untersuchten Modellen, die während der Workshops erstellt wurden, befinden sich insbesondere graphische Modelle als ikonische Bildmodelle in verschiedenen Facetten, die mehrere der Perspektiven ihres Artefakts abbilden. Neben Fotografien des Artfakts, d.h. photographischen Modellen, wurden auch teil- oder vollschematische Abbildungen u.a. auf Skizzen identifiziert.

Ikonische Bildmodelle in Form von Fotografien wurden verhältnismäßig oft erstellt. Es wurde auch versucht, Modelle mit formalen Eigenschaften wie der Hardwarekonfiguration damit abzubilden. Die Skizzen enthielten auch symbolische Elemente wie z.B. Pfeile, aber vorwiegend ikonische Abbildungen, z.B. Zeichnungen des Artefaktkorpus. Damit können sie als teil- oder vollschematische Abbildungen eingeordnet werden. Aber auch als symbolische Darstellungsmodelle einzustufende Modelle finden sich, und zwar in Form der diagrammartigen Struktur von Amici-Programmen oder dem Modelboard in MoLab, das es erlaubt Einträge anzuordnen, die gemeinsam als Darstellungsmodell den Modellierungsprozess abbilden.

Textuelle Modelle traten sowohl in den Unterkategorien umgangssprachliche Modelle als auch als formale programmiersprachliche Modelle auf. Als umgangssprachlich wurden die Texte eingeordnet, die als stichwortartige Beschreibungen, als Annotationen auf Skizzen, aber auch als Methodennamen in Amici vorkamen, wohingegen weitere in Amici eingefügte Texte, wie z.B. Pinnnummerierungen und Werte, Teil der Programmier‚sprache' sind.

Skizzen und Amici-Programme enthalten ikonische und symbolische Elemente, ergänzt durch erklärende, abundante textuelle (Unter-)Elemente. Amici kann als Darstellungsmodell (in den Unterkategorien Flussdiagramme und Schaltbilder) eingeordnet werden, enthält aber auch Elemente, die ikonischen Charakter haben und als ikonische Bildmodelle gelten können.[23]

Die mit MoLab oder dem VirtualLab erstellten Projekte wurden als Artefakt-Modelle bezeichnet. Sie sind Modelle, die das Artefakt und ggf. seinen Entstehungsprozess als Ganzes abbilden. Eine Einordnung der Artefakt-Modelle in die Kategorien der AMT kann nicht eindeutig erfolgen, da sie aus verschiedenen einzelnen Modellarten zusammen gesetzt sind.

Abbildung 7.17 fasst die Einordnung der diskutierten Modelle in die Kategorien der AMT zusammen.

[22] Das materielle Artefakt kann generell auch als Modell eingeordnet werden, z.B. in der Unterkategorie elektrotechnischer Modelle oder ggf. auch als mechanisches Modell, wurde aber in der Untersuchung nicht berücksichtigt (siehe Abschnitt 7.1.1).

[23] Insgesamt kann Amici-Programmcode auch als programmiersprachliches Modell aus einer künstlichen, formal definierten Sprache bezeichnet werden, deren Zeichen durch festgelegte graphische Symbole repräsentiert werden, die wiederum syntaktischen Regeln folgen (siehe Abschnitt 3.1.4).

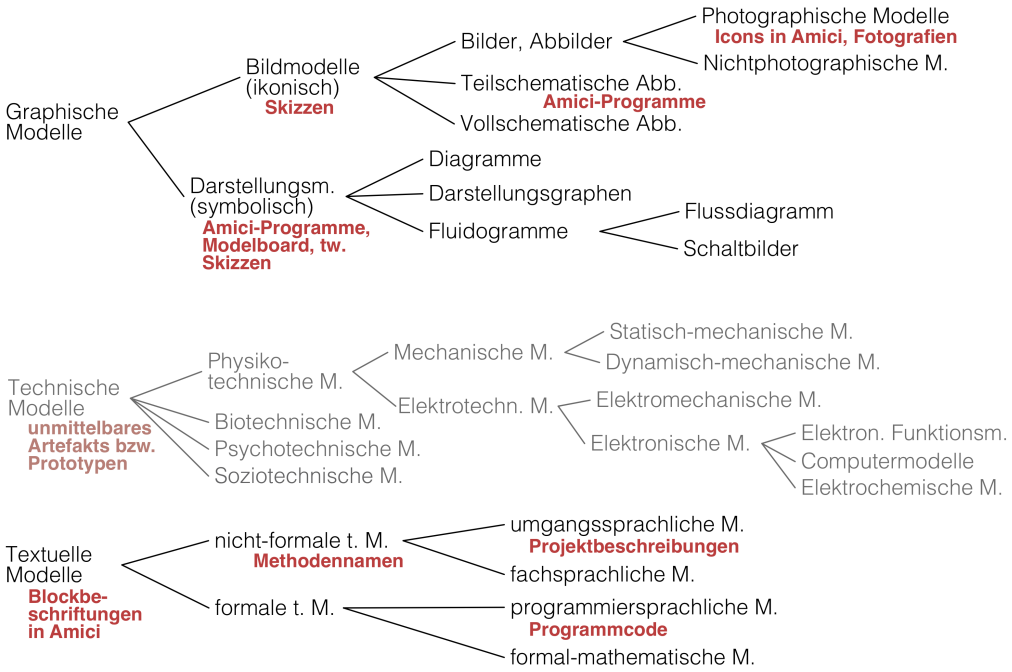


Abbildung 7.17: Mögliche Einordnung der untersuchten Modelle in AMT-Kategorien (basierend auf Abbildung 3.1)

Abgebildete Perspektiven in Modellarten

Wie in Abschnitt 7.3 bereits ausführlich dargelegt, wurde die Perspektive auf interaktives Verhalten eher in textuellen Modellen abgebildet. Die Methode des Vorspielens von Szenarien konnte dazu anregen, Interaktion mit einem Artefakt vorzumachen und auf den dabei entstandenen Fotografien anzudeuten. Wie zu erwarten war, sind Perspektiven auf die Programmierung des Artefakts vorwiegend in Programmcode enthalten. Auch Attribute der Hardwarekonfiguration fanden sich überwiegend in Amici-Code, aber auch in Form von Bildmodellen auf Fotografien und Skizzen. Modelle, die den unformalen, statischen Artefaktkorpus zeigen, wurden überwiegend in Fotografien und Skizzen als Bildmodelle dokumentiert, gelegentlich aber auch in textuellen Modellen beschrieben. Als pragmatische Merkmale wurden Intentionen und Modellsubjekte in Projektbeschreibungen erwähnt. Kontext und Modellsubjekte wurden auf Fotos, meist eher implizit, abgebildet.

Pragmatische Merkmale wurden vergleichsweise wenig abgebildet. Es ist zu vermuten, dass außerhalb von Workshops ein stärkerer Bedarf dazu gesehen wird, wie in der Fallstudie anklang.

Modelle der konkret wirkenden Seite des Programmcodes – des interaktiven Verhaltens eines Artefakts – wurden in Bildmodellen und textuellen Modellen vornehmlich angedeutet. Es bleibt zu überlegen, inwiefern das Dokumentieren von Modelle dieser immer ‚unfertigen' Perspektive notwendig ist und ob Andeutungen in Text und

Bild nicht ausreichen können. Denn wenn Programmcode und Modelle der statischen Aspekte und pragmatischer Merkmale so verständlich und nachvollziehbar dokumentiert werden, dass sie es erlauben, das Artefakt nachzubauen und selbst zu erleben, ist ein zusätzliches Abbilden von interaktivem Verhalten weniger relevant.

7.5.2 Umgang mit formalen Eigenschaften

Ein stofflich-*digitales* Artefakt zu erstellen erfordert Formalisierungsschritte. Zum einen muss das interaktive Verhalten, das dem Artefakt zugedacht ist, in einer Programmiersprache gefasst werden, zum anderen muss das Computersystem mit seinem Interface, um die Anweisungen ausführen zu können, nach bestimmten, formal-beschreibbaren Regeln spezifiziert werden. Insbesondere für Menschen, die sich bislang wenig mit Informatik beschäftigt haben, bedeutet dies eine Herausforderung. Deshalb geht die Evaluation auch der Frage nach, wie formale Eigenschaften abgebildet werden.

Für die Programmierung müssen die Modellierenden unformal wahrnehmbares Artefaktverhalten formalisieren und in einer formalen Sprache mit eindeutiger Syntax und Regeln formulieren. Um letzteres zu erleichtern, wurde bei den untersuchten Projekten die Visuelle Programmiersprache Amici eingesetzt, so dass Programme durch das Aneinanderreihen graphischer Blöcke erstellt werden konnten. Formalisierungen, die dynamische Interaktionsmöglichkeiten beschreiben, fanden sich hauptsächlich als Amici-Code und wurden außerhalb der erstellten Programme nur selten abgebildet (siehe oben). Darüber, wie die Interaktion formalisiert und in einen Algorithmus gefasst wurde, welche Zwischenschritte ggf. noch gedanklich verfolgt wurden, können die Untersuchungen wenig Auskunft geben. Teilweise fanden sich in Texten im VirtualLab Formulierungen, die ‚algorithmisch' anmutende Anweisungen enthalten,[24] oder Formulierungen, die Formalisierungen (z.B. Benennung bestimmter Werte) mit unformaler Beschreibung mischten[25]. Diese Texte wurden aber erst erstellt, nachdem das Artefakt fertiggestellt und programmiert worden war, so dass sie wenig darüber aussagen, wie die Ideen formalisiert (und algorithmisiert) wurden.

Zu den relevanten formal beschreibbaren Eigenschaften zählen nicht nur die Programmierung, sondern auch die Hardwarekonfiguration des Artefakts. Um die erdachte Interaktion zu ermöglichen, muss ein Interface technisch implementiert werden. Dazu müssen Hardwarekomponenten als Schaltkreise, die formalen Regeln folgen, mit dem programmierbaren Board verbunden werden. Deshalb interessiert hier auch, wie diese formal beschreibbaren statischen Eigenschaften des Artefakts modelliert und dokumentiert wurden, obwohl kein speziell dafür entwickeltes Werkzeug zur Verfügung gestellt wurde.

[24] Zum Beispiel Formulierungen wie „Werte die höher sind als 350".

[25] Zum Beispiel: „Er enthält einen Abstandswarner der auf Werte die höher sind als 350 [reagiert]. Der Abstandswarner schaltet den Vibrationsmotor (am Schwanz) und die LEDs (die als Augen dienen) an."

Im professionellen Kontext werden zur Beschreibung von Hardwarekonfigurationen Schaltbilder benutzt. In Dokumentationen von Makerprojekten im Internet finden sich vielfach ikonische Darstellungen, die mit Programmen wie Fritzing (siehe Abschnitt 5.3.4) erstellt wurden. In den hier durchgeführten Untersuchungen können im Nachhinein verschiedene Modellarten unterschieden werden, die benutzt wurden, um Hardwarekonfigurationen abzubilden. Es fanden sich zum einen ikonische Bildmodelle als Fotografien, zum anderen teil- oder vollschematische Bildmodelle in Form der mit textuellen Modellen erweiterten Skizzen. Auf Fotografien wurden die ausgebreiteten und prototypisch miteinander verbundenen Hardwarekomponenten festgehalten. Nachteil war, dass die Nummern und genaue Lage der Anschlüsse kaum oder nicht zu erkennen waren. Auf Skizzen der Artefakte wurde der Verlauf von Leitern und Hardwareteilen oft eingezeichnet. Sie bildeten die Art der Komponenten und ihre Lage am Artefakt ab, weniger aber, wo diese am Board angeschlossen werden müssen.

Aufgrund der Vorgaben der Amici-Umgebung (siehe Abschnitt 6.4.2) enthielten deren Programme auch Hardwareinformationen in Form von ikonischen Bildern (Bild des Sensor/Aktuators) und textuellen Elementen (der Pinnnummer).[26] Dass diese Informationen nicht nur beim Programmieren, sondern auch beim Wiederaufnehmen von Projekten nützlich sind, zeigte die Fallstudie. Dort stellte sich heraus, dass die VPL Amici ein relativ einfaches und verständliches Abbilden der formal beschreibbaren Hardwarekonfiguration ermöglicht, das ausreichende Informationen zum Wiederaufnehmen von Projekten liefern kann. Visueller Code, der mit Amici oder vergleichbaren Umgebungen erstellt wird, erleichtert also nicht nur das Programmieren, sondern erfüllt auch eine Funktion zur Dokumentation von Programm und Hardwarekonfiguration und verwendet dabei eine eher konkret anmutende, leicht verständliche Darstellung formaler Eigenschaften.

Eine Projektdokumentation, die in einem der beiden späteren Workshops mit MoLab erstellt wurde, enthielt eine ausführliche Beschreibung der Hardwarekomponenten und -anschlüsse als textuelles Modell. Dazu waren im Textfeld eines MoLab-Eintrags die Hardwarekomponenten und die entsprechenden Anschlüsse am Arduino-Board gelistet. Zu erwähnen ist, dass dieses Projekt nicht als Amici-Code, sondern in textueller Arduino-Programmiersprache programmiert wurde. Dem textuellen Code können zwar die Pinnnummern (wo eine Komponente am Board angeschlossen ist) entnommen werden. Was für ein Sensor, Schalter oder Aktuator aber genau angeschlossen ist, ist nicht ohne Weiteres erkennbar.[27] Entsprechend dokumentierten die Jugendlichen die relevanten Informationen über die Hardwarekonfiguration separat. Ob die Projektgruppe diese auch bei Verwendung von Amici zusätzlich aufgeschrie-

[26] Wobei zu bemerken ist, dass das Auswählen passender Hardware-Icons freiwillig ist und nicht den Programmablauf beeinflusst.

[27] Beim textuellen Programmieren könnten passend gewählte Variablennamen einen Hinweis darauf geben.

ben hätte, kann nicht überprüft werden. Dennoch spricht dieses Beispiel dafür, dass Amici sich gut für verständliche Hardwaremodellierung bzw. -dokumentation eignet.

Normalerweise wird Programmcode erstellt, um dem ausführenden Computer Anweisungen zu geben. Im vorliegenden Fall wurde er (auch) genutzt, um als Modell Teilaspekte des Artefakts zu dokumentieren. Als Dokumentation ist es entscheidend, dass der Programmcode übersichtlich genug ist, um schnell erfasst werden zu können, und dass die Darstellungsform verständlich für die Betrachtenden ist. Wenn Programmcode also nicht nur benutzt wird, um einen Computer zu programmieren, sondern auch als dokumentierendes Modell des Artefakts dient, so scheint visueller Programmode durch seine Kompaktheit und Übersichtlichkeit hierfür besser geeignet zu sein als textueller Code. Ähnliche Prinzipien bieten auch moderne IDEs für professionelle Entwickler_innen. So genannte ‚reverse engineering'-Funktionen erlauben es, aus textuellem Programmcode nachträglich übersichtlichere Diagramme zu erstellen.[28]

Im visuellen Amici-Programmcode ist nicht vermerkt, an welchen Stellen am Artefaktkorpus sich die Hardwarekomponenten befinden und wie sie in dieses integriert sind. Auch ein Schaltbild würde die Positionen der Hardwareteile nicht eindeutig abbilden. Die Anordnung der Hardwarekomponenten auf dem Artefakt muss daher in anderen Modellen dokumentiert werden, z.B. in Modellen, die auch die Gestalt des Artefaktkorpus zeigen. Die meisten der untersuchten Skizzen würden in dieser Hinsicht eine gute Ergänzung zum Programmcode bieten, indem sie Modelle der Schnittstelle von Hardware und Artefaktkorpus zeigen.

Zusammenfassend kann gesagt werden, dass vorwiegend ikonisch-symbolische Mischmodelle genutzt wurden, um formale Eigenschaften zu dokumentieren. Wenngleich Amici solche Abbildungsmöglichkeiten vorgibt, so wurden Hardwarekonfigurationen z.B. auch auf Skizzen abgebildet, die sowohl als ikonische Bildmodelle als auch als symbolische Darstellungsmodelle eingeordnet werden können oder eine Mischform darstellen. Formal beschreibbare oder zu beschreibende Gegebenheiten wurden durch konkrete Abbildungen oder Bezeichnungen kontextualisiert und mit Alltagsdingen oder -handlungen in Verbindung gebracht, z.B. durch Ergänzung ikonischer Bildmodelle oder durch erklärende textuelle Modelle. Es lässt sich folgern, dass das Verbinden symbolischer Darstellungsmodelle mit ikonischen Bildmodellen, die auf visuell wahrnehmbare Aspekte des Artefakts verweisen, für Amateur_innen geeignet ist, um formale Sachverhalte verständlich für sie selbst abzubilden.

7.5.3 Erkenntnisse über MoLab

Die Erprobung von MoLab wurde von der Frage geleitet, wie sich die von MoLab implementierten Anforderungen in der Praxis bewähren. An dieser Stelle werden die Ergebnisse der Untersuchungen von MoLab zusammengefasst und auf die Anforde-

[28] Zum Beispiel die Umgebung EclipseUML von Omondo: `http://www.ejb3.org` (aufgerufen am 31.07.2014)

rungen, für die sich wesentliche Erkenntnisse ergaben, bezogen (siehe auch Tabelle 7.2).

Die Webplattform MoLab wurde als ein Modellierungstool erstellt, das eine Reihe von Funktionen zur Verfügung stellt und diverse Anforderungen implementiert, die das Abbilden von Modellen unterstützen sollen. Unter anderem sollte mit MoLab in der Praxis überprüft werden, inwiefern sich schrittweises Dokumentieren (ähnlich How-tos oder Storyboards) für Projekte, die sich noch im Entstehungsprozess befinden, eignet. Wie sich in den Erprobungen zeigte, wurden nur wenige Funktionen, mit denen in MoLab Modelle erstellt werden können, genutzt. Vorwiegend war dies das Hochladen von Programmcode als separate Datei und das Einbinden von Bildern vorwiegend per Webcam. Hauptsächlich abgebildet wurden also Modelle aus den Perspektiven Programmierung, Hardwarekonfiguration, Artefaktkorpus und pragmatische Merkmale als Modellsubjekte. Modelle, die Informationen zu interaktivem Verhalten und Intentionen enthalten, wurden dagegen eher implizit erstellt. Eine Perspektivenvielfalt war damit nur beschränkt vorhanden. Die vorgegebenen Kategorien zur Perspektiventrennung wurden nicht genutzt, stattdessen wurden jeweils in einem MoLab-Eintrag Modelle verschiedener Perspektiven eingestellt. Betrachtet man alle im Rahmen der Fallstudie erstellten MoLab-Einträge zusammen, so bilden sie zusammengenommen aber doch ein recht umfangreiches Modell des Artefakts und seines Entstehungsprozesses.

Die Einbeziehung des *persönlichen* Subjektkontexts war insofern hoch, als dass sich die Modellierenden oft selbst abbildeten. Das lag vor allem an der Webcam-Funktion. Die Webcam-Integration war auch hilfreich und förderte die Praxisintegration des Tools. Dennoch überwiegt der Eindruck, dass das Dokumentieren mit dem Tool zu aufwändig ist und dass es sich als Dokumentationstool schlecht in ein konzentriertes Makervorgehen integriert. Ein Tool wie MoLab lenkt den Fokus vom eigentlichen Konstruktionsprozess ab. Demzufolge wurden Einträge auf der Plattform eher am Rande von Workshopterminen – gegen Ende und oft unter Zeitdruck – erstellt, so die Beobachtung.

Abgebildet wurde der jeweils aktuelle Stand des Projekts. Das heißt, nur Modelle, die sich seit dem letzten Mal geändert hatten, wurden dokumentiert. Da iterativ gearbeitet wurde, fand sich z.B. in fast jedem MoLab-Eintrag der aktuelle Programmcode. Die redundant abgebildeten Modelle (z.B. geänderter Code) bestätigen, dass sich das Abbilden in Schritten eher weniger eignet. Somit bestand das Modellnetz, das auf dem Modelboard sichtbar wurde, aus wenigen chronologischen Einträgen. MoLab-Einträge wurden nicht wie vorgesehen kategorisiert und auf dem Modelboard gruppiert. Allerdings wurden oft verschiedene Modelle in einem Eintrag zusammengefasst, wodurch auf diese Weise Modellschnittstellen abgebildet wurden. Zusammenhänge einzelner Modelle (z.B. Entstehung des Programms) über Termine hinweg waren aus der Übersicht nicht ersichtlich.

Die Erprobung von MoLab ergab aber auch, dass ein anderes im Workshop benutztes Tool Potenzial als Modellierungs- und Dokumentationstool hat – die Programmierumgebung Amici.

7.5.4 Potenziale Visueller Programmierumgebungen als Modellierungstools

In beiden Untersuchungen kristallisierte sich heraus, dass es sich bei den Modellen, die am ehesten erstellt und dokumentiert werden und die den meisten Informationsgehalt hinsichtlich formaler Aspekte bieten, um visuelle Amici-Programme handelt. Deshalb wird an dieser Stelle ausführlich betrachtet, welche Erkenntnisse die Evaluation über Amici als Modellierungstool liefert – auch wenn die Fallstudie sich vorwiegend an MoLab orientierte. So soll das Potenzial Visueller Programmierumgebungen als Modellierungstools ergründet werden.

Übersichtlichkeit und Nachvollziehbarkeit

Mit Visuellen Programmiersprachen (VPLs) werden Programme erstellt, die z.B. auf einer Syntax aus Blocksymbolen beruhen. Eine Abbildung eines solchen Programms ist (auch) ein graphisches Modell. Diese Modelle erinnern strukturell auch an Diagramme, wie sie z.B. im Software-Engineering-Prozess benutzt werden. Aus den visuellen Programmen generiert die Software automatisch Textcode. Der Textcode ist, zumindest bei Amici oder anderen VPLs für Arduino (siehe Abschnitt 5.3.4), für die Programmierenden parallel sichtbar. Auch kann im Textcode weiterprogrammiert werden.

Gegenüber dem Textcode hat das graphische Abbild den Vorteil, dass es reduzierter und übersichtlicher ist. Mit ‚wenigen Blicken' können relevante Eigenschaften des dynamisch-interaktiven Verhaltens erfasst werden.[29] Abbildung 7.18 zeigt einen Vergleich zwischen graphischem Amici-Code und daraus generiertem Arduino-Code.

In der Praxiserprobung von MoLab stellte sich heraus, dass Amici-Programme umfassende Modelle darstellen, die ausreichten, um das aktuelle Projekt fortzuführen. So stellten die Teilnehmenden des Workshops fest, dass sie auf das Foto, das die Hardwarekonfiguration zeigt, auch hätten verzichten können, um ihr Projekt wieder aufzubauen und weiterzuentwickeln. Denn im Amici-Programmcode konnten sie – sobald die Dateien in Amici geöffnet waren – den Hardwareaufbau ablesen durch die dort abgebildeten Pinnnummern und die Foto-Icons der jeweils angeschlossenen Sensoren und Aktuatoren.

Mit Amici wird nicht nur die Programmierung erstellt und dokumentiert, sondern auch die Hardwarekonfiguration. Grundlegende Informationen für die Wiederaufnahme eines (eigenen) Projekts sind im graphischen Abbild des Amici-Programmcodes

[29] Ein sehr kurzes Textprogramm mag u.U. auch schnell erfassbar sein, allerdings enthält es viele zusätzliche Zeilen wie z.B. Variablendeklarationen, die in Amici versteckt sind. Auch müssen Variablennamen bewusst formuliert werden, um Bezug zum Artefakt herzustellen.

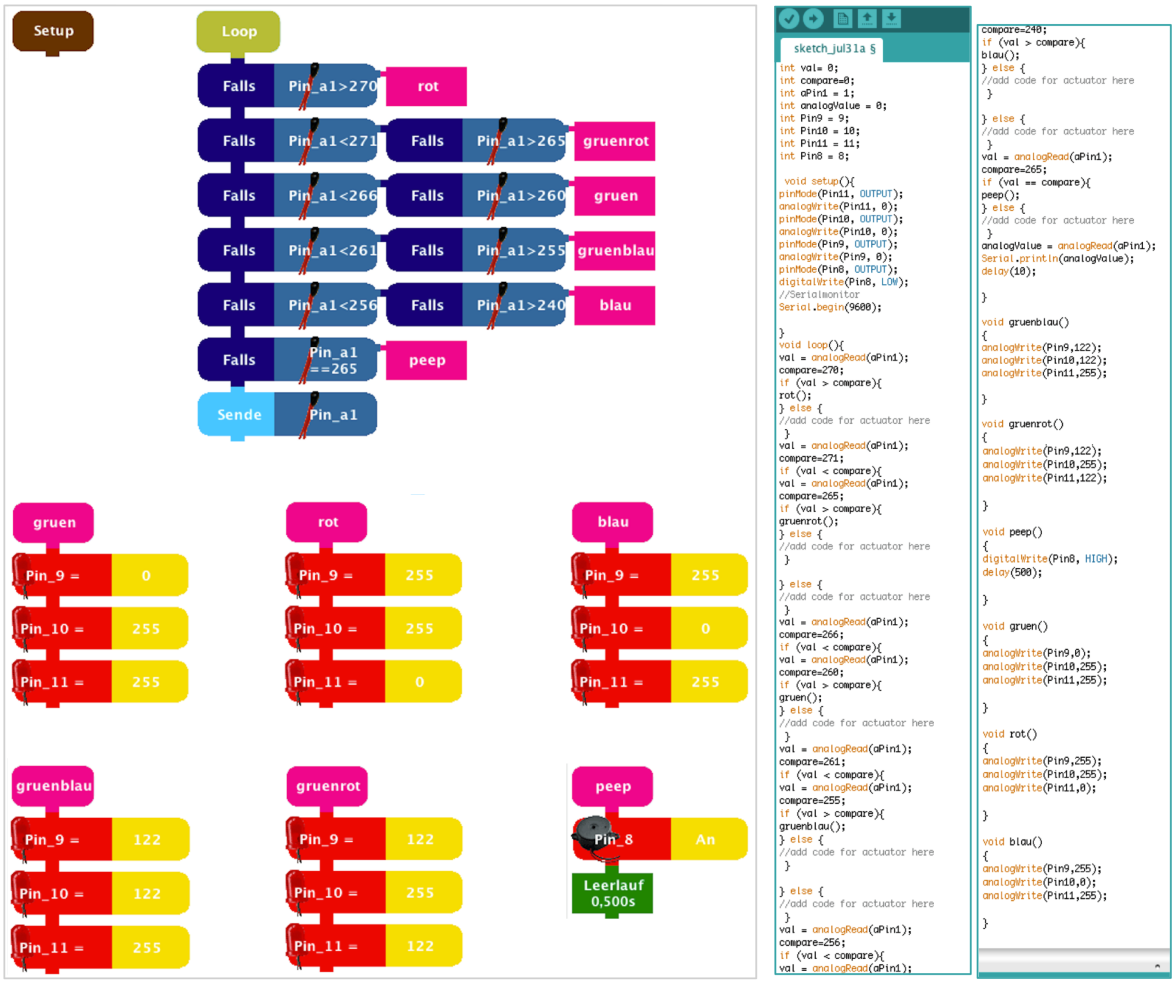


Abbildung 7.18: Visueller Amici-Programmcode (links) und daraus erzeugter textueller Arduino-Code (rechts)

enthalten, ohne sie zusätzlich dokumentieren zu müssen. Damit erweist sich die Visuelle Programmierumgebung nicht nur für die Modellierung des Programms (mit einer Perspektive auf formale, dynamische Artefakteigenschaften) als gut geeignet, sondern auch zur Modellierung und Dokumentation statisch-formaler Eigenschaften.

Amici-Modelle zeichnen sich somit durch eine übersichtliche Darstellung formaler Artefaktaspekte aus, die das Nachvollziehen von Projekten begünstigt. Ermöglicht wird dies durch die Weiterentwicklungen, die in Amici implementiert wurden (siehe Abschnitt 6.4.2). Sie tragen nicht nur – wie ursprünglich gedacht – zum Verständnis der Programme bei, sondern bringen Vorteile hinsichtlich einer umfassenderen Modellierung verschiedener Artefaktaspekte und deren Schnittstellen. So wird das Abbild des visuellen Programmcodes auch zu einer relativ aufschlussreichen Dokumentation des programmierten Artefakts.

Verschiedene Perspektiven

Neben Hardwareinformationen wurden weitere Modellperspektiven in Amici-Programmen entdeckt, die zur Nachvollziehbarkeit des Artefakts beitragen. Die Modellanalyse ergab, dass dieser Programmcode nicht nur einen Algorithmus und die Hardwarekonfiguration abbildet, sondern auch auf andere Aspekte des Artefakts verweisen kann. Dateinamen und Methodennamen gaben Hinweise auf unformales interaktives Verhalten und/oder Aussehen des Artefaktkorpus oder deuteten kontextuelle pragmatische Merkmale an, die z.B. auch Informationen über den persönlichen Kontext geben können. Amici-Code kann somit Perspektiven auf Programmierung, Hardwarekonfiguration und teilweise auch auf das unformale interaktive Verhalten enthalten und Rückschlüsse auf die Gestalt des Artefaktkorpus zulassen. Zugleich hilft das Einbeziehen der Hardware den Programmierenden, eine Verbindung zwischen Programm und stofflichem Artefakt herzustellen. Darüber hinaus zeigt Amici eine für Anfänger_innen noch abstraktere Ebene – den textuellen Programmcode, der automatisch generiert wird. Damit vermag die Programmierumgebung konkrete, unformale Seiten des Artefakts mit abstrakten, formalen Repräsentationen – den Blöcken – zu verknüpfen. Somit können Modelle in Amici nicht nur verschiedene Teilsysteme eines Artefakts abbilden, sondern auch Perspektiven auf verschiedene Abstraktionsstufen des Artefakts aufweisen und miteinander verbinden. Amici erlaubt es also – wenngleich nicht Modelle aller Perspektiven erstellt wurden – verschiedene Aspekte eines Artefakts abzudecken und damit die Anforderung an Perspektivenvielfalt stärker zu erfüllen als gedacht. Auch Modellschnittstellen werden dadurch nicht nur zwischen Hardware und Programmierung, visuellem und textuellem Code geschaffen, sondern ansatzweise auch zu unformalen, in Methodennamen abgebildeten Eigenschaften (siehe auch Tabelle 7.2).

Eignung von Amici als Modellierungstool

Amici ist also auch ein Modellierungstool, das es erlaubt, verschiedene Modelle übersichtlich und nachvollziehbar zu dokumentieren. Der visuelle Code hat in dieser Hinsicht Vorteile. Er reduziert die Komplexität der Programmiersprache, macht das Programm rasch überblickbar und erfassbar – und eignet sich dadurch gut zu Dokumentationszwecken. Des Weiteren beinhalten Amici-Programme Modelle verschiedener Perspektiven auf das Artefakt, die es umfassend genug darstellen, um dieses zumindest selbst weiterzumodellieren. Die Programmierumgebung wird zum Modellierungstool, mit dem Modelle erstellt werden, die austauschbar sind, den aktuellen Projektzustand zeigen und unterschiedliche Eigenschaften des Artefakts abbilden.

Ein Programm ist ein wesentlicher Teil eines stofflich-digitalen Artefakts und somit ist die Programmierung fester Bestandteil des kreativen Konstruktionsprozesses. Im untersuchten Fall hielten die Modellierenden für sie relevante Eigenschaften während der Programmierung fest und nutzten die visuelle Darstellung als Dokumentation, anhand derer sie später ihr Artefakt weitermodellierten. Das Modell des graphischen Programmcodes nahm eine zentrale Funktion im Modellierungsprozess ein. Durch seinen Nutzen zur Programmierung ist Amici, im Gegensatz zu MoLab, fest in den Modellierungsprozess integriert. Deshalb ist es ratsam Modellierungs- und Dokumentationstools auf Grundlage Visueller Programmierumgebungen zu entwickeln.

Da visueller Code übersichtlicher gestaltet und für Anfänger_innen leichter nachvollziehbar ist als textueller Code, können VPLs als Basis für Projektdokumentationen möglicherweise einen weiteren Vorteil bieten. Angehende Maker, die ein Projekt nachbauen wollen, könnten eher dazu motiviert sein, den übermittelten Programmcode zu verstehen und ihn zu ändern, anstatt ihn bloß zu kopieren (so wie es durch How-tos meist anregt wird, siehe Abschnitt 2.5.4) und sich damit intensiver mit der Programmierbarkeit Digitaler Medien auseinandersetzen. Da sich die gängigen VPLs (siehe Abschnitt 5.3.4) zur Programmierung von Arduino-Mikrocontrollern in ihrer Syntax sehr ähneln, muss auch nicht befürchtet werden, dass die Nutzung einer bestimmte VPL den Code nur für eine kleine Nutzergruppe verständlich macht.

7.5.5 Rückblick auf die Anforderungen

Entsprechend der Ergebnisse der Modellanalysen und der Fallstudie mit MoLab sind die in Abschnitt 6.1.4 erstellten Anforderungen an Modellierungstools rückblickend zu hinterfragen und anzupassen.

Besonders wichtig, um Modellieren bzw. Dokumentieren von Modellen anzuregen, scheint die möglichst nahtlose *Praxisintegration* in den Modellierungsprozess zu sein, indem angeknüpft wird an Tools, die für das Umsetzen des Projekts selbst erforderlich sind. Modellierungswerkzeuge sollten in den Konstruktionsprozess eingebunden werden, statt als ein externes Tool den Konstruktionsprozess zu unterbrechen. *Flexi-*

bilität im Sinne von vielen Funktionen scheint nicht notwendig oder eher hinderlich zu sein, da sich dadurch die Übersichtlichkeit der Werkzeuge und Modelle verringerte. Flexibilität in Verbindung mit *Einfachheit* wie die Einbindung von Bildern mittels Webcam erwies sich als praktikabel. Insbesondere konkrete bildliche Modelle, die mit eher bricoleurartigem oder designtypischem Herangehen assoziiert werden, können so ohne viel Aufwand angeregt und erstellt werden. Auch dass die überschaubaren und simplen Amici-Programme verhältnismäßig viele Informationen über ein Projekt enthalten und bevorzugt zur Dokumentation genutzt wurden, bestätigt die Forderung nach einfachen, nicht überladenen Tools.

Persönlicher Kontext wurde oft durch Abbildung oder Erwähnung der Modellsubjekte eingebunden. Der Kontext des Artefakts wurde eher zufällig abgebildet. Am ehesten scheinen dafür mit der Webcam erstellte Fotos geeignet zu sein, oder – wie im VirtualLab gefunden – auch Texte.

Die Anforderung *Allgemeinverständlichkeit* war ein grundlegendes Auswahlkriterium für die benutzten und entwickelten Tools und wurde in der vorliegenden Untersuchung nicht explizit überprüft. Dass die Workshopteilnehmer_innen in der Lage waren, aus graphischem Amici-Code unmittelbar ihr Projekt zu rekonstruieren, deutet daraufhin, dass diese Art der Repräsentation auch für noch relativ unerfahrene Maker schnell erfassbar ist.

Die Tools VirtualLab und MoLab vermochten das Abbilden des Artefakts aus verschiedenen Perspektiven (als *Perspektivenvielfalt*) zu fördern, aber auch die VPL Amici deckte mehr Perspektiven als erwartet ab. Alle Tools ermöglichten es auch konkrete, unformale Perspektiven auf das zu modellierende Artefakt zu integrieren. Amici stellt neben dem graphischen Programm automatisch den Textcode dar und damit noch eine zusätzliche, abstraktere Perspektive auf die Programmierung. Eine *Perspektivtrennung* als eine Zuordnung von Modellen zu Kategorien wurde von den Nutzenden von MoLab nicht angenommen. Auch die analysierten Modelle nahmen meist keine isolierten Perspektiven ein, sondern mischten z.B. formale mit unformalen Eigenschaften. Für Makerprojekte kleinen Umfangs sollte vielmehr eine umfassende Abbildung des aktuellen Zustands des Artefakts im Mittelpunkt stehen. Als eine solche zentrale Abbildung geeignet erwies sich visueller Amici-Code. Dieses Modell könnte je nach Bedarf durch verschiedene weitere Modelle, die nicht strikt vorgegebenen Perspektiven zugeordnet werden müssen, ergänzt werden.

Auch das bewusste Verbinden von Modellen zu *Modellnetzen* als Artefakt-Modell oder Modellierungsprozess schien in der Praxis wenig interessant zu sein. Relevant für die Modellierenden war immer die aktuellste, funktionierende Version ihres Projekts. Interessant wäre zu untersuchen, wie Dritte die Projekte anhand der abgebildeten Modelle nachbauen und ob nicht dann eine Anordnung, aus der ersichtlich ist was in welcher Reihenfolge gemacht wurde, mehr Bedeutung bekommt.

	Allgemeinverständlichkeit	**Persönlicher Kontext**	**Flexibilität**	**Praxis -integration**	**Einfachheit (Erstellen von Modellen)**	**Perspektivenvielfalt**	**Perspektivtrennung**	**Modellnetze/ -folgen**	**Modellschnittstellen/ -übergänge**
Amici (2012)	weniger umgangsspr.; visueller Code und Hardware-Icons aber übersichtlicher als textueller Code	*in Methodennamen angedeutet;* visueller Bezug zur Realwelt über Hardware-Icons	niedrig: keine anderen Modelle außer vorgegebenen Elementen möglich	hoch Programmierung muss sowieso gemacht werden	Programmierung relativ einfach und unkompliziert; aber keine anderen Modelle möglich	*vis. Programm mit Hardware-Informationen und abstr. Textcode; weitere Persp. in Methodennamen*	Fokus auf Programmierung; Informationen zu Hardwarekonfig. im vis. Programmcode	nicht möglich	*automatisch zwischen Hardware & Programmierung, vis. Code & Textcode; zu Interaktion tw. über Methodennamen*
MoLab	Bilder, Text, Storyboarding	*hoch (Einbezug von Fotos, freie Anordnung): Modellsubjekte bildeten sich oft ab*	hoch: versch. Dateiformate, versch. Anordnungsmöglichkeiten	*zu aufwändig; Webcam-Integration aber hilfreich.*	gering: erfordert Login und Transfer von Programm sowie Arbeitsunterbrechungen	*haupts. abgebildet wurden: Programm u. Hardware-Info durch Amici-Code; Subjekte und Gestalt durch Fotos. Interaktion u. Intentionen weniger*	*vorgegebene Unterscheidung wird nicht genutzt*	*nur chronologische Kette: pro Termin wird max. ein Eintrag erstellt*	*möglich durch eigene Gruppierung von Einträgen, wird aber nicht angenommen*

Tabelle 7.2: Erfüllung der Anforderungen anhand der Evaluationsergebnisse (neue Erkenntnisse kursiv)

Im Gegensatz zur Perspektivtrennung gewannen *Modellschnittstellen* an Bedeutung. So bildet Amici z.B. Hardwarekonfiguration und Programm gemeinsam ab, was auch für die Umsetzung notwendig ist, denn ohne die richtig angeschlossenen Hardwarekomponenten kann das Programm nicht den gewünschten Effekt haben. Auch die Hardware und der Artefaktkorpus wurden gemeinsam abgebildet, um z.B. auf Skizzen anzugeben, wo Hardwarekomponenten zu platzieren sind. Das Abbilden von Modellschnittstellen ist also wichtig beim Making, um notwendige Zusammenhänge darzustellen und ein Artefakt nachvollziehbar abzubilden. Auch so kann offensichtlich zur Bewältigung der Komplexität stofflich-digitaler Artefakte beigetragen werden.

Rückblickend werden neben der Allgemeinverständlichkeit die Anforderungen Persönlicher Kontext, Praxisintegration, Einfachheit, Perspektivenvielfalt und Modellschnittstellen als besonders relevant angesehen, während die übrigen (insbesondere Perspektivtrennung) eher zu vernachlässigen sind. Tabelle 7.2 zeigt einen Überblick, inwiefern MoLab und Amici die Anforderungen nach den Erkenntnissen, die die Evaluation gebracht hat, erfüllen.

7.5.6 Empfehlungen für Modellierungstools

Neben den verbliebenen Anforderungen können aus den Erkenntnissen der Evaluation Gestaltungsempfehlungen abgeleitet werden, welche Elemente oder Funktionen ein Modellierungstool enthalten sollte, so dass wesentliche Aspekte des Artefakts nach-

vollziehbar abgebildet werden können (siehe Fragen in Abschnitt 7.1). Zusammengefasst werden sie als „Modellieren durch Konstruieren“, „Bildbasierte Modelle“ und „Teilautomatisiertes Dokumentieren“.

Modellieren durch Konstruieren. Beim Making sollten der Spaß am Konstruieren und der kreative, iterative Konstruktionsprozess im Vordergrund stehen. Zu bedenken ist, wie das Dokumentieren in die Konstruktionstätigkeit integriert werden kann, ohne diese unnötig zu stören. Dabei sollte erwogen werden, wie Modelle, die unbewusst schon im Konstruktionsprozess entstehen, zu Dokumentationszwecken genutzt werden können. Eine Visuelle Programmierumgebung wie Amici, die das Programm und die daran anknüpfende Hardwarekonfiguration auch für Anfänger_innen verständlich und übersichtlich abbildet, bietet einen geeigneten Ausgangspunkt für die Entwicklung eines Modellierungstools. Denn die Programmierung ist durch ihre essentielle Funktion zur Implementierung des Artefakts bereits im Konstruktionsprozess verankert und bei iterativem Vorgehen meistens präsent. Es empfiehlt sich, ausgehend von einem solchen visuellen Programm, verschiedene zusammengehörige Aspekte des Artefakts gemeinsam darzustellen, um ein integratives Modellieren zu berücksichtigen. Der aktuelle Stand und die aktuelle Tätigkeit sollten dabei im Vordergrund stehen, ältere Iterationen eher ausgeblendet werden. Auch hat sich gezeigt, dass in kleinsten Textbausteinen Informationen wie z.B. Intentionen, die etwas über das Projekt verraten, mitgeteilt werden. Bausteine, die zum Konstruieren oder Programmieren benötigt werden, könnten z.B. um kleine Textfelder (als Bildunterschriften, Methodennamen, Projekttitel etc.) erweitert werden.

Bildbasierte Modelle. Bildmodelle haben sich als geeignetes Medium erwiesen, um diverse Aspekte eines Artefakts zu dokumentieren. Sie erlauben ein relativ konkretes Darstellen abstrakter Sachverhalte und eignen sich auch für formalere Eigenschaften, indem diese in gemischten formal-unformalen Modellen abgebildet werden. Um den Konstruktionsprozess wenig zu stören, ist eine einfache, unkomplizierte Möglichkeit Fotos mit einer angeschlossenen Kamera zu erstellen und einzubinden sinnvoll und kann das Dokumentieren von Modellen fördern. Eine integrierte Kamera oder Webcam ist aber nicht nur geeignet, um Fotos des Artefakts zu erstellen. Auch können damit externe Modelle wie Skizzen, Interaktionsabläufe u.a. abfotografiert werden. Visuelle Programme sollten nicht nur als codierte Datei (wie XML im Falle von Amici), sondern zusätzlich als Screenshot des Programms gespeichert werden, so dass diese Modelle auch ohne die passende Programmierumgebung angesehen werden können.

Teilautomatisiertes Dokumentieren. Modelle, die etwas über das Artefakt aussagen, werden im Konstruktionsprozess bereits erstellt, wie z.B. ein Programm, oder entstehen auch unbeabsichtigt. Da Dokumentieren eine eher unbeliebte Tätigkeit

ist, sollte erwogen werden, wie bestimmte Dokumentationsabläufe vom System automatisch unterstützt oder übernommen werden können. Ein Modellierungstool kann die Benutzenden automatisch bei bestimmten Schritten daran erinnern, Modelle einzubinden – z.B. das Erstellen von Fotos vorschlagen oder sie auffordern, aussagekräftige Projektnamen oder Projektziele anzugeben. Eine automatische Versionierung und Speicherung von entscheidenden Projektschritten ist zu empfehlen, so dass Außenstehende oder Beteiligte den Modellierungsprozess nachvollziehen können. Auch sollte ein System das Weiterverwenden von Modellen für andere Dokumentationsformate unterstützen. So könnten einzelne Modelle, Artefakt-Modelle oder auch der Modellierungsprozess in gängigen Formaten (z.B. als Bilder, Texte, HTML) exportiert werden, so dass die Modellierenden sie anschließend zum Beispiel für das Erstellen eines How-Tos nutzen können.

Jedoch sollten jegliche Automatisierungen gründlich abgewogen werden, um das Verstehen des Artefakts und beteiligter Sachverhalte nicht zu unterbinden oder zu behindern, aber auch, um – im Sinne der ersten Empfehlung – den Konstruktionsprozess nicht mehr als notwendig zu stören.

Einordnung der Empfehlungen und ergänzende Anmerkungen

Die genannten Empfehlungen sind darauf ausgerichtet, Tools zu entwickeln, die junge Maker_innen beim Modellieren, Dokumentieren und Wiederaufnehmen von Projekten unterstützen und dabei die selbstkonstruierten stofflich-digitalen Artefakte als Digitale Medien erfahrbar machen.

An mehreren Stellen in dieser Arbeit klang an, dass beim Making neue Sachverhalte auf informelle Weise gelernt werden und dass Reflexion ein wichtiger Teil des Handelns und des Lernens darstellt (siehe Abschnitte 2.5.3 und 4.3.1). Modellierungstools sollten solche Reflexionsphasen nicht – etwa durch Automatisierungen – unterbinden. Nach den bisherigen Erfahrungen wird aber auch davon abgeraten, diese von einem Softwaretool intervenierend vorzugeben. Vielmehr sollte darauf vertraut werden, dass sich auch während des aktiven iterierenden Umsetzungsprozesses Gelegenheiten bieten das eigene Vorgehen zu reflektieren (wie die aktuelle Arbeit von Dittert (2015) zeigt) und dass beim Wiederaufnehmen von Projekten anhand von Dokumentationen Situationen entstehen, in denen neue Perspektiven auf das bisherige Vorgehen eingenommen werden.

7.5.7 Konsequenz der Ergebnisse: Amici+

Anhand der revidierten Anforderungen und der Empfehlungen wurde eine Erweiterung von Amici konzipiert und implementiert – Amici+. Damit wird exemplarisch aufgezeigt, wie die gegebenen Empfehlungen in ein Modellierungstool einfließen kön-

	Allgemeinverständlichkeit	**Persönlicher Kontext**	Flexibilität	**Praxisintegration**	**Einfachheit (Erstellen von Modellen)**	**Perspektivenvielfalt**	Modellnetze/ -folgen	**Modellschnittstellen/ -übergänge**
Amici+	visueller Code mit Hardware-Bildern **ergänzt um eigene Fotos und Beschreibungen**	**zusätzlich Möglichkeit des Einbezugs über eigene Fotos sowie des persönlichen Ziels**	**erlaubt nun auch externe Fotos; visueller Amici-Code allerdings kein Standard; erfordert Installation**	hoch Programmierung muss sowieso gemacht werden	**visueller Code ist zentrales Element; erfordert kein zusätzliches externes Tool, mit Webcam Fotos einfach einzubinden**	Programm mit Hardware-Informationen; **Fotos und Notizen für div. Aspekte (z.B. für Artefaktgestalt); Frage nach Ziel**	**nur aktueller Stand sichtbar, dafür Projektversionen automatisch abgespeichert; Einbinden von Bildfolgen möglich**	Hardware/ Programmierung; vis./text. Code; **Fotos u. Notizen z.B. der Artefaktgestalt in Programmieroberfläche**

Tabelle 7.3: Berücksichtigung der bewährten Anforderungen durch Amici+ (Neuerungen hervorgehoben; Anforderung Perspektivtrennung weggelassen)

nen und wie eine VPL zu einem umfassenderen Modellierungstool erweitert werden kann.

Konzept

Für Amici+ wurde Amici entsprechend der oben aufgestellten Empfehlungen erweitert. Damit erfüllt es auch die bewährten Anforderungen besser als Amici (siehe Tabelle 7.3), die Grundlage für die Gestaltungsempfehlungen waren.

Eine Schwachstelle von Amici als Modellierungstool ist eine eingeschränkte Perspektivenvielfalt. Wie in der Analyse der mit Amici erstellten Programme in Abschnitt 7.3 festgestellt wurde, werden mit dieser VPL pragmatische Modellmerkmale (z.B. Zweck des Projekts) und unformale statische Eigenschaften des Artefakts vergleichsweise selten abgebildet. Dies ist nicht weiter verwunderlich, da Amici ursprünglich als Programmierumgebung konzipiert war. So gibt es wenige bzw. eher indirekte Abbildungsmöglichkeiten für pragmatische Merkmale, interaktives Verhalten und Artefaktgestalt. Außerdem speichert Amici ein Programm als XML-codierte Datei, d.h. diese muss mit der Software geöffnet werden, damit die enthaltenen Modelle betrachtet werden können.

Programmzentriert: Im Zentrum von Amici+ (siehe Abbildung 7.19) steht das Amici-Programm, das, wie oben beschrieben, die Perspektiven Programm und Hardware abbildet, sowie ggf. noch weitere eher unformale Informationen mittels der Methodennamen. Über dem Programmierfenster werden Fotos (mit Notizen) angezeigt. Auch kann das Projektziel eingetragen werden. Somit stellt Amici+ ein Projekt in seinem aktuellen Zustand aus mehreren Perspektiven auf derselben Oberfläche dar. Zusammenhänge – Schnittstellen – zwischen den aktuellen Modellen werden so ersichtlich.

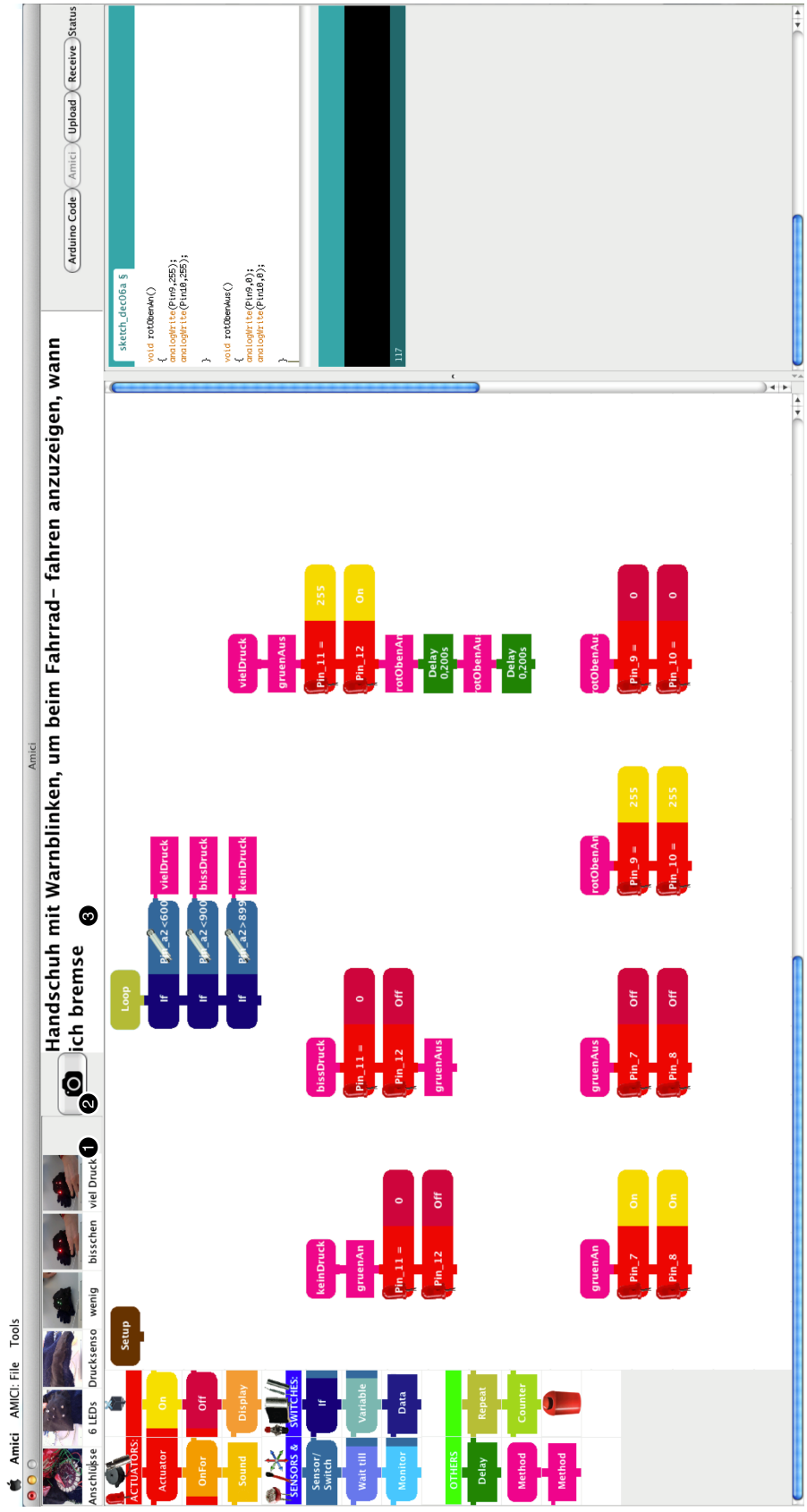


Abbildung 7.19: Oberfläche von Amici+ mit Beispielprojekt „Fahrradhandschuh“ (Erläuterung der mit Nummerierung gekennzeichneten Funktionen im Text)

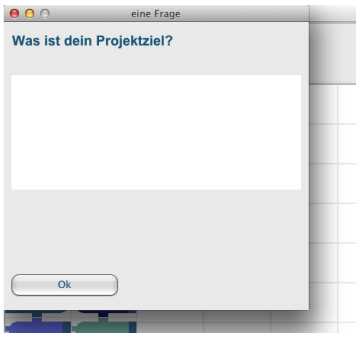


Abbildung 7.20: Aufforderung Projektziel zu nennen

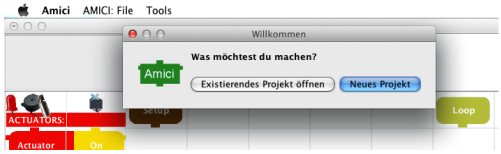


Abbildung 7.21: Aufforderung bereits begonnenes Projekt weiterzuführen

Perspektivenerweiterung durch Bildmodelle: Wie in den Analysen festgestellt wurde, sind ikonische Bildmodelle ein geeignetes Mittel, um auch formalere Eigenschaften im Kontext des Selbermachens stofflich-digitaler Artefakte abzubilden. Deshalb ergänzt Amici+ das Programmfenster um weitere Modelle, vorwiegend Bildmodelle, die zusätzliche Perspektiven auf das Artefakt ermöglichen. Die Fotos werden als eine Reihe kleiner Bilder angeordnet und können mit kurzen Notizen versehen werden (siehe Annotation 1 in Abbildung 7.19). Durch Anklicken wird ein Bild vergrößert angezeigt. Die Bildfolge kann als Storyboard genutzt werden und eignet sich damit auch dazu, Interaktionsabläufe abzubilden.

Fotos können über eine Webcam-Anbindung direkt erstellt und in die aktuelle Ansicht integriert werden. So lassen sich auch externe Bildmodelle (z.B. Skizzen) einbinden. Durch Klicken auf den Kamera-Button (siehe Annotation 2 in Abbildung 7.19) öffnet sich ein übergeordnetes Fenster mit der Kamerasicht, von wo aus die Fotos aufgenommen werden können.

Auto-Dokumentation: Amici+ automatisiert bestimmte Abläufe. So können mehr Modelldokumentationen entstehen, ohne dass der Konstruktionsprozess allzu sehr unterbrochen werden muss.

Intentionen, die mit einem Projekt verfolgt werden, wurden zwar wenig dokumentiert, im Interview nach der MoLab-Erprobung wurden sie aber als wichtig beschrieben. Wenn Intentionen mitgeteilt wurden, so geschah dies am ehesten über Text. Das gleiche gilt für das interaktive Verhalten des Artefakts. Um diese Aspekte festzuhalten, erscheint beim Anlegen eines neuen Projekts eine Aufforderung das Projektziel zu benennen (siehe Abbildung 7.20). Durch die Frage nach dem Ziel darf erwartet werden, dass auch pragmatische Merkmale genannt werden. Wird das Projektziel so formuliert, dass es die Attribute der Interaktionsperspektive abbildet, so kann es auch als sinnvolle Vorlage während des Programmierens dienen, an der sich die Programmierenden orientieren können (siehe Annotation 3 in Abbildung 7.19).

In Workshops wurden Dokumentationen meist am Ende eines Termins erstellt. Amici+ knüpft daran an. Wollen die Nutzenden die Software beenden, werden sie daran erinnert, ein Foto des Artefakts zu erstellen.

Das aktuelle Projekt wird regelmäßig automatisch gespeichert. So können sich die Benutzenden auf die Konstruktionsarbeit konzentrieren, es entsteht aber auch kein Datenverlust, weil das Abspeichern vergessen wird. Amici+-Projekte werden in einem Ordner als ‚Projektpaket' mit Datum und Uhrzeit versehen gespeichert. Darin enthalten sind Programmcode, Screenshots (s.u.), Fotos und Notizen. Auf diesen Ordner kann auch direkt über das Dateisystem (ohne die Software Amici) zugegriffen werden, um z.B. Fotos oder die XML-Programmdatei für anderweitige Verwendung (z.B. für ein How-to) zu entnehmen.

Um das Wiederaufnehmen früherer Projekte zu fördern, wird beim Start von Amici+ nun gefragt, ob ein bestehendes Projekt geöffnet oder ein neues angelegt werden soll (siehe Abbildung 7.21). Damit soll unterstützt werden, dass angefangene Projektpakete, in denen schon Fotos des Projekts gesammelt wurden, weiter benutzt werden.

Bei der Erprobung von MoLab wurde festgestellt, dass Amici-Programme viel Aussagekraft haben. Die XML-Datei, in der Amici-Programme abgespeichert und als Dateianhang in MoLab und das VirtualLab hochgeladen werden, können in dieser Form aber nicht auf der Website angesehen werden, sondern müssen erst wieder mit Amici geöffnet werden. Nicht nur Code ist relevant, ähnlich wichtig ist auch die daraus ablesbare Hardwarekonfiguration, z.B. um zu erkennen, wo Aktuatoren und Sensoren angeschlossen werden müssen, damit das Programm wirken kann. Deshalb legt Amici+ automatisch Screenshots des graphischen Programms im Projektordner ab. Dazu wird jedes Mal, wenn das Programm auf das Arduino-Board geladen wird, ein Screenshot des Programms in einem gängigen Bildformat (PNG) abgespeichert. Diese Bilder von Amici-Programmen können dem Projektordner entnommen und für externe Projektdokumentationen (z.B. Instructables, MoLab oder VirtualLab) verwendet werden. Da regelmäßig Screenshots erstellt werden, können diese später auch genutzt werden, um Anhand der Bilder Entstehung und Änderung der Programmierung nachzuverfolgen, ohne einzelne Programmdateien mit einer Amici-Software öffnen zu müssen.

Umsetzung

Für Amici+ wurde Amici (siehe Abschnitt 6.4.2) erweitert und basiert dementsprechend auf der Software Arduino[30] (aktuell Version 1.0) und ist in Java[31] programmiert (JDK1.5). Außerdem wurde für die Webcam-Anbindung die Open-Source-Bibliothek „Webcam Capture" [32] benutzt.

[30] `http://arduino.cc/en/Main/Software` (aufgerufen am 27.11.2013)
[31] `http://www.oracle.com/technetwork/java/index.html` (aufgerufen am 27.11.2013)
[32] `http://webcam-capture.sarxos.pl/` (aufgerufen am 27.11.2013)

Das VPL-basierte Modellierungstool Amici+ ist das praktische Ergebnis dieser Arbeit. Ob es die ihm zugrundeliegenden Annahmen in der Praxis erfüllt, wurde nicht evaluiert.

7.6 Zusammenfassung und Schlussfolgerungen

Dieses Kapitel hatte zum Ziel, theoretische Annahmen und Anforderungen an Modellierungstools in der Praxis zu überprüfen. Dazu wurden zum einen Modelle empirisch untersucht, die junge Amateur_innen im Rahmen von TechKreativ-Workshops erstellten. Zum anderen wurde im Rahmen einer Fallstudie beschrieben und ausgewertet, wie das Modellierungstool MoLab und die VPL Amici in einem Workshop benutzt wurden. Damit konnten auch die mit MoLab und Amici erstellten Modelle über die Analyse hinaus kontextualisiert und die Eignung der Tools in der Praxis überprüft werden. Das Vorgehen bei beiden Auswertungen lehnte sich an die qualitative Inhaltsanalyse von Mayring (2010) an. Da die Modellanalyse auch quantitative Aussagen bringen sollte, wurde die skalierend-strukturierende Inhaltsanalyse gewählt, wohingegen für die Auswertung der Fallstudie die inhaltlich-strukturierende Variante gewählt wurde.

Mit der Modellanalyse wurde untersucht, welche Modelle erstellt werden, welche Perspektiven diese abbilden und wie sie aus theoretischer Sicht einzuordnen sind. Dazu wurden Skizzen verschiedener Art, Beschreibungen und Fotos des Vorspielens von Szenarien, Einträge aus dem VirtualLab und darin enthaltene Amici-Programme sowie die in MoLab erstellten Einträge (Artefakt-Modelle) und deren einzelne Elemente untersucht. Die Auswertung ergab, dass die meisten Modelle – auch solche mit Attributen formaler Eigenschaften – als ikonische Bildmodelle verschiedener Ausprägungen sowie – im Falle von visuellem Programmcode – als Darstellungsmodelle zu kategorisieren sind. Mit den verschiedenen Medien Bild, Text und VPL wurden unterschiedliche Perspektiven abgebildet, wobei eine Modellart meist mehrere Perspektiven enthielt. In Kombination sind diese Modelle in der Lage, ein stofflich-digitales Artefakt bzw. dessen aktuellen Zustand relativ umfassend mit den Attributen, die für das Weitermodellieren oder Nachvollziehen relevant sind, darzustellen.

Das Tool MoLab, das verschiedene Modellierungsmöglichkeiten anbietet und ein kleinteiliges Abbilden von Modellierungsprozessen ermöglicht, erwies sich als weniger geeignet. Es unterbricht den eigentlichen Konstruktionsprozess und bietet zu viele Funktionen an, von denen nur wenige benötigt werden. Die Auswertung zeigte stattdessen ein besonderes Potenzial Visueller Programmiersprachen wie Amici für das Modellieren verschiedener Perspektiven eines stofflich-digitalen Artefakts. Für die Modellierenden waren die damit erstellten Dokumentationen ausreichend, um ihre Projekte wieder aufzubauen, da hier auch die Hardwarekonfiguration mit wenigen Blicken erfassbar ist. Darüber hinaus werden auch Attribute anderer Perspektiven in textuellen Modellen (z.B. Methodennamen) integriert, wie bei der Modellanalyse festzustellen

war. Eine Programmierumgebung ist ein wesentlicher Teil des Modellierungsprozesses informatischer Systeme und damit auch des Selbermachens stofflich-digitaler Artefakte. Somit ist sie in den Prozess des Making integriert. Die Ergebnisse legen nahe, dass Modellierungstools an Visuelle Programmierumgebungen anknüpfen sollten.

Anhand der Evaluationsergebnisse wurden die Anforderungen an Modellierungstools auf Allgemeinverständlichkeit, Persönlicher Kontext, Praxisintegration, Einfachheit, Perspektivenvielfalt und Modellschnittstellen reduziert und um die Gestaltungsempfehlungen „Modellieren durch Konstruieren“, „Bildbasierte Modelle“ und „Teilautomatisiertes Dokumentieren“ ergänzt. Um diese exemplarisch umzusetzen, wurde Amici entsprechend um Einbindungsmöglichkeiten eigener Bildmodelle und Automatisierungsfunktionen erweitert. Das neue Modellierungstool Amici+ ist das praktische Ergebnis dieser Arbeit, wie junge angehende Maker_innen beim Erstellen von Modellen stofflich-digitaler Artefakte unterstützt werden können, ihre sich im Entstehungsprozess befindenden Artefakte umfassend und nachvollziehbar abzubilden. Die theoretischen Ergebnisse dieser Arbeit werden in Kapitel 8 zusammengefasst.

Kapitel 8

Zusammenfassung und Schlussfolgerungen

8.1 Zusammenfassung

Diese Arbeit begann mit einem Einblick in die Praxis des Selbermachens stofflich-digitaler Artefakte, das – eingebettet in eine Makerbewegung – die Herausforderung enthält, sich mit abstrakten Konzepten auseinanderzusetzen und diese verständlich für sich und andere zu dokumentieren. Digitale Medien sind ein charakterisierendes Element dieser Makerbewegung, indem sie als Kommunikationsmedium, als Tools und/oder als Zielobjekte untrennbar Bestandteil des Selbermachens von Dingen geworden sind und zugleich neue Möglichkeiten des informellen Lernens für Amateur_innen eröffnen. Insbesondere Technologien wie Arduino ermöglichen auch Laien das Erstellen stofflich-digitaler Artefakte. Als ein wesentlicher Bestandteil dieser Makerbewegung und auch des Lernens beim ‚Making' wurde das Tauschen von Projektdokumentationen und -zwischenständen identifiziert. Dabei haben sich How-tos als Format für Projektdokumentationen etabliert. Ein allgemeinverständliches, nachvollziehbares Abbilden abstrakter Sachverhalte und von Projektzwischenständen unterstützen How-tos jedoch nur unzureichend.

Das dritte Kapitel entwickelte eine theoretische Grundlage, um Making und die dabei involvierten Artefakte als Modellbildung zu beschreiben. Stofflich-digitale Artefakte sind informatische Systeme, die (u.a.) konzeptuell modelliert werden und als Modelle gelten können. Die Allgemeine Modelltheorie von Stachowiak (1973) wurde als theoretischer Zugang gewählt. Sie berücksichtigt auch pragmatische Eigenschaften von Modellen wie Subjektivität und Zweck. Das formale Modellieren von unformaler Wirklichkeit führt jedoch zu Zielkonflikten, die sich teilweise aus dem konzeptuellen Vorgehen begründen.

Da Mensch-Computer-Interaktion in soziale Kontexte eingebettet ist, können stofflich-digitale Artefakte nur unzureichend formal-konzeptuell modelliert werden. So hat sich die HCI zu einem interdisziplinären Gebiet entwickelt, das ‚unformale' Herangehensweisen einschließt. In Kapitel 4 wurden alternative Vorgehensweisen aus dem HCI-Kontext – insbesondere Designgebieten – herangezogen. Designtypisches Vorgehen wurde zu diesem Zweck von ingenieurtypischem oder informatischem Vorgehen abgegrenzt. Gekennzeichnet ist ersteres durch ein reflektierendes Iterieren an konkreten Prototypen und kann ebenfalls als Modellbildungsprozess eingeordnet werden. Das Konzept des Modellierens stofflich-digitaler Artefakte wurde so um unformale Herangehensweisen und Modelle erweitert. Ansätze aus dem Mechatronikkontext verweisen auf die Komplexität des Entwickelns stofflich-digitaler mechanischer Artefakte und zeigen, dass sich konkrete Modellierungszugänge bzw. Repräsentationen nutzen lassen, um abstrakte Modelle verständlicher zu machen.

In Kapitel 5 wurden Tools und Methoden betrachtet, die vorwiegend für Designer_innen oder Amateur_innen zur Modellierung informatischer Systeme entwickelt wurden. Diese verfolgen unformale Herangehensweisen (z.B. Storyboards und Howtos) und/oder erlauben das Modellieren formaler Sachverhalte, indem konkrete Repräsentationen mit abstrakten verbunden werden. Auch wurden die Tools VirtualLab und Amici, die solche Konzepte umsetzen, eingeführt. Die existierenden Tools beschränken sich jedoch auf das Modellieren und Abbilden bestimmter Aspekte eines stofflich-digitalen Artefakts oder dessen Modellierungsprozesses. Gestaltungsempfehlungen für Tools im Makerkontext betonen u.a., flexibel mit unterschiedlichen bevorzugten Herangehensweise von Nutzenden umzugehen.

Die verschiedenen theoretischen Perspektiven wurden in Kapitel 6 zueinander in Beziehung gesetzt und auf Making als Modellbildung übertragen. Making wurde als ein eher designartiger, kreativer, iterierender und nicht-linearer Modellierungsprozess eingeordnet, bei dem eine Vielfalt von Modellen entstehen. Modelle dienen dabei als Reflexionsobjekte, anhand derer Ideen und Projekte entwickelt werden. In dokumentierter Form dienen sie dem Austausch und Lernen voneinander. Das setzt voraus, dass Modelldokumentationen Zusammenhänge zwischen formalen Modellen und unformaler Erscheinung als auch pragmatische Eigenschaften darlegen. Identifiziert wurden Perspektiven, aus denen stofflich-digitale Artefakte modelliert werden (können): als Modelle der Programmierung, des interaktiven Verhaltens, der Hardwarekonfiguration oder des Artefaktkorpus. Sie stellen die zu modellierenden Teilsysteme oder Aspekte eines Projekts dar und werden bestimmt von pragmatischen Merkmalen. Für Modellierungstools für angehende Maker_innen wurden die Anforderungen Allgemeinverständlichkeit, Persönlicher Kontext, Flexibilität, Praxisintegration, Einfachheit, Perspektivenvielfalt, Perspektivtrennung, Modellnetze und Modellschnittstellen abgeleitet. Um sie in die Praxis zu übertragen, wurden mit dem Vorspielen von Szenarien und dem Entwickeln von Skizzenvorlagen analoge Modellierungsmethoden konzi-

piert. Weiterentwicklungen der Visuellen Programmierumgebung Amici wurden implementiert, um die Einbindung der stofflichen Hardwarekonfiguration und die Sichtbarkeit des textuellen Codes, der sich hinter der graphischen Oberfläche verbirgt, zu verbessern. Um verschiedenartige Modelle und Modellierungsprozesse dokumentieren zu können, wurde die Webplattform MoLab entwickelt, die How-to- oder Storyboard-ähnliche Strukturen benutzt und eine breite Palette an Funktionen zum Erstellen oder Einbinden von kleinteiligen Modellen anbietet.

Im siebten Kapitel wurden Modelle, die mit MoLab, Amici, dem VirtualLab als auch mit analogen Methoden in TechKreativ-Workshops erstellt worden waren, hinsichtlich abgebildeter Perspektiven und Modellarten inhaltlich analysiert. Bei den analysierten Modellen spielten ikonische Bildmodelle zur Abbildung fast aller Perspektiven eine wichtige Rolle. Ergänzt und kontextualisiert wurden die Ergebnisse durch die Evaluation der Benutzung von MoLab im Rahmen einer Fallstudie. Dabei erwies sich MoLab als schlecht in den Modellierungsprozess integrierbar, wohingegen die Visuelle Programmierumgebung Amici bisher unbeachtete Qualitäten erkennen ließ. Daraus wird gefolgert, dass eine VPL wie Amici ein geeignetes Tool zum Erstellen von Modellen stofflich-digitaler Artefakte durch Amateur_innen darstellen kann, wenn diese um einfache Einbindungsmöglichkeiten von Fotos und die Möglichkeit textueller Ergänzungen und Teilautomatisierungen erweitert wird. Entsprechend wurden die Anforderungen revidiert und die Gestaltungsempfehlungen „Modellieren durch Konstruieren“, „Bildbasierte Modelle“ und „Teilautomatisiertes Dokumentieren“ formuliert. Um diese exemplarisch umzusetzen, wurde Amici+ konzipiert und implementiert.

8.2 Methodendiskussion

Diese Arbeit greift mit der Allgemeinen Modelltheorie von Stachowiak (1973) einen theoretischen Ansatz auf, anhand dessen das Entwickeln stofflich-digitaler Artefakte als Modellbildung beschrieben wird. Dieser Zugang ermöglicht ein systematisch-analytisches Vorgehen und ein Fokussieren auf externe Repräsentationen – Modelle – die während des Konstruktionsprozesses entstehen. Die Allgemeingültigkeit der AMT erlaubt es, die Modellierung stofflich-digitaler Artefakte nicht nur als informatisch-konzeptuelle Modellierung zu betrachten, sondern auch Modellierungsprozesse anderer Fachgebiete mit unkonventionelleren und unformaleren Herangehensweisen einzubeziehen. Dabei berücksichtigt die AMT auch pragmatische Merkmale von Modellen und somit deren Konstruiertheit. Des Weiteren stellt die AMT Kategoriensysteme zur Verfügung, die es ermöglichen, verschiedenste Arten externer Repräsentationen, die in einem Modellbildungsprozess entstehen, systematisch zu analysieren. Es lässt sich schlussfolgern, dass sich das theoretische Konstrukt der AMT als Ausgangslage eignet, um Making theoretisch zu charakterisieren. Auch bietet die AMT eine Basis zur Eva-

luation vielfältiger externer Repräsentationen, um insbesondere strukturelle Aspekte zu untersuchen.

Mittels der inhaltlich-strukturierenden Analyse von Modellen konnten Modellperspektiven und Modellarten identifiziert werden sowie ein quantitativer Überblick geschaffen werden, mit welchen Modellarten die jeweiligen Artefaktaspekte dargestellt werden. So ein systematischer Ansatz birgt die Gefahr, den eigentlichen Prozess des Modellierens und das Handeln der Modellierenden zu vernachlässigen und die pragmatischen Merkmale der Modelle zu übersehen. Deshalb wurde die Modellanalyse um Untersuchungen in der Praxis ergänzt. Erst durch ihre Kontextualisierung mittels der Fallstudie werden pragmatische Dimensionen ausführlich berücksichtigt. So konnten Erwägungen und Handlungen der Teilnehmenden, die sich nicht in den dekontextualisierten Modellen abbilden, bedacht werden. Die Auswertungsmethode orientierte sich zwar an einem theoriegeleiteten Kategoriensystem, war aber offen genug für neue Beobachtungen, so dass z.B. Hinweise auf die Bedeutung der Schnittstellen zwischen Perspektiven und die Potenziale von VPLs gefunden werden konnten.

Neben dem theoretischen Herangehen wurde Wert darauf gelegt, die Tools anhand von Rückmeldungen von Nutzenden und Beobachtungen iterativ weiterzuentwickeln. Mittels des Ansatzes des design-based research konnte die jeweilige Situationsbezogenheit berücksichtigt werden, auch wenn sich die Ergebnisse dadurch nur bedingt über diese Studie hinaus validieren und generalisieren lassen. Hätte diese Arbeit vorwiegend ein erfolgreiches Tool als Ziel gehabt, wäre ein Nutzer_innen-orientierter Entwicklungsprozess im Sinne eines Partizipativen Designs bereits in der konzeptuellen Phase angebracht gewesen. So hätte typischen Zielkonflikten informatischer Modellierung, die auch die technischen Entwicklungen in dieser Arbeit beeinflusst haben, besser begegnet oder vorgebeugt werden können, z.B. verschiedene Annahmen und Interpretationen von Toolfunktionen seitens der Entwickelnden und der Nutzenden.

Die geringe Anzahl der Teilnehmenden und die Rahmenbedingungen der Workshops schränken die Generalisierbarkeit der Ergebnisse ein. Dafür ermöglicht die Untersuchung anhand einer Fallstudie in teilnehmender Beobachtung einen eingehenden Einblick, wie Jugendliche beim dokumentierenden Modellieren vorgehen. Da davon auszugehen ist, dass es sich bei Maker_innen um eine sehr diverse Zielgruppe handelt und sie sich in ihren individuellen Vorerfahrungen, persönlichen Interessen und Motivation unterscheiden, wird es immer schwierig sein, zu allgemeingültigen Aussagen zu kommen. Es wurde nicht untersucht, ob Menschen, die nicht an einem der Workshops beteiligt waren, die mit den Modellierungstools dokumentierten Modelle nachvollziehen und entsprechend ein Artefakt nachbauen könnten. Stattdessen wurde mehr Wert darauf gelegt, sich dem Thema strukturell über Modelle zu nähern als auch Modellierungsprozesse eingehend zu begleiten, um beide charakterisieren zu können. Die Ergebnisse dieser Arbeit und die resultierenden Empfehlungen geben wertvolle Hinweise, wie junge Amateur_innen, die stofflich-digitale Artefakte erstellen, aber

noch wenig Erfahrung im Umgang mit den beteiligten Technologien haben, beim Dokumentieren ihrer Projekte unterstützt werden können.

8.3 Ergebnisse und Schlussfolgerungen

Die vorliegende Arbeit zeigt, wie das Selbermachen stofflich-digitaler Artefakte als Modellbildungsprozess beschrieben werden kann. Making ist demnach ein nicht-linearer kreativer Modellierungsprozess, den ein Iterieren anhand vielfältiger formaler und unformaler Modelle kennzeichnet. Ein wesentliches Modell dieses Konstruktionsprozesses ist das stofflich-digitale Artefakt selbst. Möglich wurde diese modellorientierte Beschreibung, indem das Selbermachen stofflich-digitaler Artefakte nicht nur im Sinne klassischer Informatik als Modellbildung betrachtet wurde. Ausgehend von der Allgemeinen Modelltheorie von Stachowiak (1973) wurden auch alternative, nicht-informatische Vorgehensweisen beim Modellieren von Artefakten, wie sie in Designbereichen vorzufinden sind, einbezogen. Dadurch gelten auch iterierendes, intuitives kreatives Vorgehen als Modellieren und gegenständliche, unformale Objekte als Modelle. Somit handelt es sich beim Selbermachen stofflich-digitaler Artefakte um einen Modellierungsprozess, der vielfältige Herangehensweisen und Modellarten einschließt. Die Fallstudie ergänzt und bestätigt die theoretischen Erkenntnisse, indem sie zeigt, dass die Teilnehmenden keine aufeinanderfolgenden, einzelnen Modelle des Systems erstellen, sondern dieselben Modelle und das Artefakt regelmäßig wieder aufgreifen und bearbeiten. Die Lernerfahrungen, die die Jugendlichen während des Modellierungsprozesses machen, beeinflussen die Ausgestaltung des Projektziels. Die meisten der dokumentierten Modelle entstehen als Teil der Konstruktion und weniger mit planerischer Absicht. Sie bilden den Zustand des konstruierten Artefakts oder Teile dessen ab und können somit auch zu Dokumentationszwecken genutzt werden.

Diese Arbeit gibt Aufschluss darüber, welche Arten von Modellen junge Amateur_innen erstellen und wie sie dabei mit formalen Eigenschaften umgehen. Denn auch wenn das Vorgehen beim Making als eher unformal beschrieben wird, müssen doch Sachverhalte formalisiert werden, um ein stofflich-digitales Artefakt zu erstellen. Unter den analysierten Modellen finden sich insbesondere graphische Modelle – vorwiegend als ikonische Bildmodelle – in verschiedenen Ausprägungen, die mehrere Aspekte des Artefakts abbilden. Bildmodelle und graphischer Programmcode werden vornehmlich zum Abbilden der Programmierung, des Artefaktkorpus und der Hardwarekonfiguration genutzt. Um formale Eigenschaften darzustellen, werden symbolische Bildmodelle mit ikonische Bildmodellen kombiniert. Auch werden textuelle Modelle benutzt, um formale Eigenschaften mit umgangssprachlichen Formulierungen zu ergänzen. So werden Zusammenhänge zwischen gegenständlichem Artefakt und abstraktem Sachverhalt hergestellt und Modellperspektiven erweitert. Wie die Modellanalyse ergab, werden Bildmodelle, insbesondere Fotografien, von den Modellierenden

bevorzugt, auch wenn alternative Abbildungsmöglichkeiten (z.B. Text) zur Verfügung stehen.

Da auch abstrakte Sachverhalte mittels Bildmodellen dargestellt werden, kann gefolgert werden, dass diese für die Zielgruppe von Nutzen sein können, um auch formalere Eigenschaften in gemischten formal-unformalen Modellen für sich verständlich und nachvollziehbar abzubilden. Bildmodelle erweisen sich damit als geeignetes Medium, um diverse Aspekte eines Artefakts und deren Schnittstellen zu dokumentieren. Zu schlussfolgern ist, dass durch die Integration konkreter gegenständlicher Abbildungen formale Aspekte verständlicher dokumentiert werden können. Dies bezieht sich nicht nur auf das Entwickeln von Interfaces, sondern auch darauf, wie junge Amateur_innen selbst Modelle erstellen und somit Eigenschaften eines stofflich-digitalen Artefakts abbilden. Darüber *wie* Formalisierungsschritte vollzogen wurden, bzw. ob dieser kognitive Prozess besser zu unterstützen wäre, können die Untersuchungen keine Auskunft geben.

Neben modelltheoretischen Erkenntnissen liefert diese Arbeit Aufschluss, welche Modellierungswerkzeuge sich eignen, um Maker_innen beim Dokumentieren von Modellen stofflich-digitaler Artefakte zu unterstützen. Die Ergebnisse geben Hinweise bezogen auf das möglichst umfassende und nachvollziehbare Abbilden von Artefaktaspekten mit verschiedenen Modellierungsmethoden. Sie zeigen, wie Tools gestaltet sein sollten, die sich in den Konstruktionsprozess integrieren und amateurgerechte Modellierungsfunktionen bereitstellen.

Das umfangreiche prozessorientierte Modellierungstool MoLab, das neu entwickelt wurde, erwies sich als nicht optimal geeignet. Es wurde wenig und hauptsächlich zum Abspeichern von Programmcode und Fotos genutzt, wie Modellanalyse und Fallstudie ergaben. Verschiedene Aspekte des Artefakts wurden abgebildet, jedoch nicht so umfangreich, wie es möglich wäre. Wie bereits bemerkt wurde, wird während des Selbermachens sehr iterativ und kleinschrittig vorgegangen. Damit ist eine lineare Modellierungs- und Dokumentationsstruktur mit in sich geschlossenen Schritten, wie sie auch von How-tos vorgegeben ist, während eines Konstruktionsprozesses weniger passend. MoLab verfügt deshalb über kleingliedrige Strukturierungsmöglichkeiten für einzelne (Teil-)Modelle ohne linearen Zusammenhang. Doch auch diese werden wenig genutzt, wie die Auswertungen von Modellen und Fallstudie zeigen. Generell ist eine externe Webplattform als Modellierungstool eher für das Dokumentieren eines fertigen Artefakts oder dessen Teilsystemen geeignet, so wie es auch in How-tos oder im VirtualLab möglich ist. Die Untersuchungen mit MoLab bestätigten aber auch, dass das Dokumentieren und Abbilden von Modellen als Fotos während des Prozesses gut anwendbar ist. Dabei erwies sich die direkte Anbindung einer Kamera an das Modellierungstool als sehr nützlich.

Eher unerwartet deckt diese Arbeit Potenziale Visueller Programmierumgebungen als Modellierungstools auf. Sie zeigt, dass sich visueller Programmcode nicht nur zum

Programmieren von Computersystemen eignet. Durch ihre relativ übersichtliche Darstellung können diese Modelle bzw. ihre Repräsentationsformate auch zur Dokumentation von Projekten genutzt werden und dabei helfen, Aufbau und Wirken stofflich-digitaler Artefakte nachzuvollziehen.

So ergab die Evaluation der Fallstudie, dass junge Amateur_innen in den Amici-Modellen unmittelbar eine Verbindung zwischen Programm und ihrem Artefakt herstellen und den Projektaufbau daraus ablesen konnten. Durch die Integration ikonischer Bildmodelle der Hardwarekonfiguration im Blockcode wird das Programm kontextualisiert und erscheint ‚konkreter'. Wie die Ergebnisse weiter zeigen, zeichnet sich visueller Amici-Programmcode durch eine Übersichtlichkeit aus, die ihn für Dokumentationszwecke prädestiniert. Durch den konkreten Bezug und die Blockstruktur, die unnötige Informationen versteckt, sind wesentliche Merkmale des Projekts schnell erfassbar, so dass angehende Maker_innen ihr Projekt zügig anhand dieser Vorlage wieder aufbauen konnten. Vergleichbare Vorteile können generell für Visuelle Programmierumgebungen gefolgert werden. Auch verbindet Amici verschiedene Modellperspektiven mit teilweise verschiedenen Abstraktionsstufen: Das stoffliche Artefakt bzw. dessen Hardwarekonfiguration und sein programmiertes Verhalten werden durch die graphischen Blöcke miteinander verbunden. Die graphischen Blöcke wiederum werden in Zusammenhang mit dem noch abstrakteren textuellen Code dargestellt. Allerdings spielte letztere Verbindung im untersuchten Fall keine wichtige Rolle, ist bei fortgeschritteneren Projekten aber denkbar.

Amici zeichnete sich auch durch die Integration verschiedener Perspektiven aus, die über die Programmierung hinausreichen. Die Modellanalyse ergab, dass Modelle verschiedener Teilaspekte des Artefakts in visuellem Programmcode bereits implizit enthalten sind, z.B. durch Methodennamen. Unformale und pragmatische Perspektiven werden darin angedeutet. So können Artefakte mit der Programmierumgebung umfassender als gedacht abgebildet werden. Um das Modellieren weiterer Artefaktaspekte – insbesondere unformaler Modelle und pragmatischer Merkmale – zu fördern, können in der Weiterentwicklung Amici+ auch fotografische Bildmodelle und zusätzliche textuelle Modelle erstellt werden.

Ein entscheidender Vorteil einer Programmierumgebung als Modellierungstool ist, dass sie wie selbstverständlich in den Konstruktionsprozess integriert ist. Durch das iterative Vorgehen beim Making wird das Programm immer wieder herangezogen und verändert, wie die Fallstudie zeigt. So ist die Programmierumgebung die meiste Zeit sowieso geöffnet. Geschlussfolgert wird, dass sich das Dokumentieren von Modellen dem Konstruktionsprozess unterordnen bzw. in diesen integrieren sollte. Modelle, die während der Konstruktion entstehen, sind stärker zu berücksichtigen und sollten für Dokumentationszwecke genutzt werden im Sinne eines Modellierens durch Konstruieren.

Visuelle Programmierumgebungen haben somit als Basis für Modellierungstools großes Potenzial und Qualitäten, um verschiedene Aspekte von stofflich-digitalen Artefakten aus verschiedenen Perspektiven abzubilden, die für Dokumentationen genutzt werden können und die auch für Amateur_innen verständlich und nachvollziehbar sind.

Wesentliche Ergebnisse dieser Arbeit wurden in den Empfehlungen „Modellieren durch Konstruieren“, „Bildbasierte Modelle“ und „Teilautomatisiertes Dokumentieren“ ausgedrückt und mit dem neuen VPL-basierten Modellierungstool Amici+ implementiert. Sie werden als essentiell angesehen für Tools, die junge Maker_innen darin unterstützen wollen, Modelle ihrer stofflich-digitalen Artefakte zu erstellen und zu dokumentieren.

8.4 Beitrag

Diese Arbeit befasste sich mit zwei aktuellen Themen aus dem Bereich der HCI, die während der Laufzeit dieser Arbeit noch weiter an Bedeutung gewonnen haben: Makerbewegung als gesellschaftliches Phänomen und Physical Computing als zukunftsweisende Technologie.

Der Begriff Makerbewegung wurde bislang in der Literatur sehr wage gebraucht und bezieht sich meist generell auf Menschen, die Dinge selber machen, ohne näher zu bestimmen, was diese ausmacht und was daran neu ist. Diese Arbeit liefert eine Darstellung der Makerbewegung und eine Definition, die Digitale Medien als essentiellen Teil identifiziert – sowohl als Werkzeug, Kommunikationsmedium als auch als Zielartefakt.

Es wurde ein modelltheoretischer Ansatz entwickelt, um Making als Modellbildung beschreiben zu können und dabei der Tatsache gerecht zu werden, dass – trotz des Entwickelns eines informatischen Systems – das Vorgehen eher unformal und auch informell erscheint. Das Konzept der Modellierung informatischer Systeme wurde dazu um eine interdisziplinäre Sichtweise auf Modelle erweitert. Durch das Einbeziehen designassoziierten Modellierens wird die Interdisziplinarität der HCI berücksichtigt, die Designzugänge nutzt, um unformale menschliche Aspekte zu modellieren. Mit dem vorliegenden Ansatz werden erstmals Makeraktivitäten als Modellierung betrachtet und Modelle und ihre Rolle dabei im Sinne der AMT beschrieben. Der Ansatz lässt sich verwenden, um auch Modellierungstätigkeiten, die mit neuen Technologien für digitale Fabrikation in Makerkontexte Einzug halten, zu untersuchen. Durch die Vielfalt der Modelle, die an stofflich-digitalen Artefakten beteiligt sind, ist dieses Konzept als theoretisches Framework für weitere Untersuchungen nutzbar.

Ausgehend von theoretischen Überlegungen wurde anhand der Modelltheorie ein Analyse-Framework entwickelt. Dieses wurde benutzt, um Modelle, die junge angehende Maker_innen erstellt hatten, empirisch zu untersuchen. So liefert diese Arbeit

durch einen systematischen Ansatz einen strukturellen Einblick, welche Eigenschaften stofflich-digitaler Artefakte junge Menschen mit wenig Vorerfahrung mit und ohne Modellierungstools abbilden. Die Untersuchung orientierte sich dabei weitest möglich an den Repräsentationen, die junge Amateur_innen selbst wählten und erstellten. Dieses Analyse-Framework kann auch mit anderen Forschungsschwerpunkten genutzt werden, um Eigenschaften externer Repräsentationen und Abstraktionsleistungen zu untersuchen, z.B. um mehr über Lernprozesse zu erfahren oder darüber, wie sich Selbermachende abstrakte Sachverhalte erschließen.

Praktische Ergebnisse dieser Arbeit sind die zwei neuen einsatzfähigen Modellierungstools MoLab und Amici+. Das webbasierte Tool MoLab stellt einen neuen Ansatz dar, Projektschritte nicht-linear und kleinschrittig darzustellen und zeigt auf, dass eine direkte Einbindungsmöglichkeit fotografischer Bildmodelle förderlich ist. Mit der Idee, Visuelle Programmierumgebungen zu umfassenden Modellierungs- und Dokumentationstools zu erweitern, wurde ein Ansatz entwickelt, der sich am Konstruktionsprozess orientiert und die vielfältigen Modelle verschiedener Perspektiven, die mit einer VPL entstehen, einbezieht. Eine Visuelle Programmiersprache wie Amici ist relativ verständlich für junge Menschen und ermöglicht ihnen, eine Verbindung zwischen abstraktem Programm und stofflichem Artefakt herzustellen. Übersichtlicher graphischer Code trägt zur Nachvollziehbarkeit von Projekten bei, so dass sich diese Darstellungsweise insbesondere für Dokumentationszwecke nutzen lässt. Die vorliegenden Konzepte für Modellierungstools berücksichtigen zum einen, welche Arten von Repräsentationen geeignet sind, um auch Modelle formaler Aspekte für und von der Zielgruppe verständlich abzubilden. Zum anderen zeigen sie, wie Tools den speziellen Charakter von Modellbildungsprozessen beim Making einbeziehen können. Durch die Implementierung der Tools MoLab und Amici+ kommen die theoretischen Erkenntnisse letztendlich nicht nur der Wissenschaft, sondern auch angehenden Maker_innen zu Gute.

8.5 Ausblick

Wie Amici+ sich in der Praxis – sowohl in Workshops als auch außerhalb – bewährt, ist noch zu untersuchen und entsprechende Anpassungen sind vorzunehmen.

Die vorliegende Arbeit konzentriert sich auf die Modelle, die beim Making entstehen, und auf für die Zielgruppe geeignete Abbildungsmöglichkeiten. Kollaborative Aspekte waren nicht Gegenstand der Untersuchungen, auch wenn entsprechende Funktionen in MoLab zur Verfügung stehen. Jedoch ist das Veröffentlichen und Teilen von Projektdokumentationen ein wesentlicher Bestandteil der Makerbewegung. Um Amici+ hierfür nutzen zu können, sollte das Tool um eine Netzwerkanbindung und Schnittstelle zu einer Webplattform erweitert werden, so dass Projektdokumentationen dort unkompliziert hochgeladen und mit anderen geteilt werden können. Mit MoLab

steht eine technische Infrastruktur bereit, die entsprechend ausgebaut werden kann. Wichtig wäre bei zukünftigen Entwicklungen, Nutzende stärker einzubeziehen, insbesondere auch Menschen, die außerhalb von Workshops Amici+ benutzen möchten.

Es bleibt offen, ob Maker_innen, die nicht in Workshops eingebunden sind, sondern getrennt voneinander zu Hause Projekte erstellen und weitermodellieren, die entwickelten Modellierungstools anders benutzen würden. Weitere Untersuchungen könnten Aufschluss darüber geben. Auch wäre interessant, eingehend zu untersuchen, inwiefern die Tools dazu beitragen können, dass die Modellierenden etwas über Digitale Medien lernen, aber auch andere Fähigkeiten erwerben.

Während diese Arbeit entstand, haben moderne Fabrikationstechnologien wie 3D-Drucker und Lasercutter an Bedeutung gewonnen und sind – nicht zuletzt durch FabLabs – zugänglich für viele geworden. Sie ermöglichen es insbesondere, den Korpus eines Artefakts durch 2D- oder 3D-Modellierung virtuell zu erstellen und anschließend auszuschneiden bzw. plastisch auszudrucken. Dadurch verändern sich die Modellarten, die beim Konstruieren eines stofflich-digitalen Artefakts beteiligt sind. Auch zum Erstellen des Artefaktkorpus, der in dieser Arbeit als unformal modelliert angenommen wurde, sind dann Formalisierungsschritte notwendig. Schon jetzt erstellen fortgeschrittene Maker_innen mit Programmen wie Fritzing Modelle für eigene Hardwareplatinen und lassen diese produzieren. Hierbei entstehen neue Modelle auf formaler Grundlage. Das bedeutet zum einen, dass virtuelle Modelle dieser statischen Aspekte des Artefakts auch digital dokumentiert und weitergegeben werden können. Andererseits muss für diese Arten von Modellen überlegt werden, ob und wie Formalisierungsschritte und verschieden abstrakte Repräsentationen entsprechend verständlich und nachvollziehbar dargestellt werden können. Dazu ließe sich der theoretische Ansatz dieser Arbeit nutzen und entsprechend weiterentwickeln.

Diese Arbeit konzentrierte sich darauf, Modellierungstools als Software zu entwickeln, so dass diese in die Praxis integrierbar und für möglichst viele Maker_innen zugänglich gemacht werden können. Durch moderne Fabrikationstechnologien wird es in Zukunft selbstverständlich sein, auch Hardwaresysteme selbst zu Hause oder in Werkstätten wie FabLabs z.B. mit PCB-Milling-Maschinen herzustellen. So könnten auch Modellierungs- und Dokumentationssysteme entwickelt werden, die materielle Komponenten enthalten, z.B. spezielle Hardware, die ihre Konfiguration selbst dokumentieren kann oder spezielle Aufbauten, die eine Kamera enthalten und automatisch das Geschehen dokumentieren. Auch haptische Modellierungs-Interfaces, die konkrete Zugänge zu abstrakten Modellen und entsprechende Übergänge erleichtern, könnten integriert werden. Die digitalen Baupläne für solche Systeme könnten kostenlos zur Verfügung gestellt werden, so dass Amateur_innen diese mit digitalen Fabrikationsmaschinen günstig nachbauen können.

Diese Arbeit liefert Ergebnisse, mit welchen Repräsentationen Nutzende stofflich-digitale Artefakte modellieren und dokumentieren und wie diese Abläufe in die Praxis

integriert werden können. Diese Ergebnisse lassen sich auch bei der Entwicklung von Learning-Analytics-Systemen nutzen. So wird im aktuellen EU-Forschungsprojekt PELARS[1], an dem die Autorin beteiligt ist, erforscht, wie Arduino-Baukästen in Learning-Analytics-Systeme integriert werden können, um den Nutzenden Hilfestellungen beim Erstellen von Schaltkreisen und beim Programmieren geben zu können. Benötigt werden dabei Modelle der Konstruktionstätigkeiten, die ein solches System auswerten kann. Das Wissen um die Art und Struktur von Modellen bzw. externen Repräsentationen, die Lernende selbst erstellen, und wie das Erstellen solcher Modelle gefördert werden kann, kann für dieses Forschungsprojekt und die Gestaltung des Learning-Analytics-Systems hilfreich sein.

Weitere aktuelle Forschungsaufrufe, wie das kürzlich ausgeschriebene Horizon-2020-Programm der EU zu Smart Cyber Physical Systems[2] zeigen, dass es sich bei der Modellierung stofflich-digitaler Artefakte um aktuelle, zukunftsweisende Technologie handelt. Anwendungsfelder sind z.B. smart homes, Internet-of-things- oder Quantified-self-Applikationen (z.B. der Trend zu neuen Fitness- und Gesundheits-Apps). Amateur_innen an solchen Entwicklungen teilhaben zu lassen, aber auch insbesondere jungen Menschen diese Technologien näherzubringen, scheint daher sowohl aus medienpädagogischer als auch aus gesellschaftlich-partizipativer Sicht relevanter denn je. So können die Ansätze und Tools dieser Arbeit entsprechend weiterentwickelt und angewandt werden, um Nutzende und Stakeholder gezielt in die Konzeption solcher Systeme im Sinne partizipativer Entwicklung einzubeziehen oder um Amateur_innen das Erstellen eigener Anpassungen und Weiterentwicklungen bestehender Systeme zu ermöglichen.

[1] Practice-based Experiental Learning Analytics Research and Support `http://www.pelars.eu` (aufgerufen am 21.09.2014)

[2] ICT 2014 - Information and Communications Technologies: Smart Cyber-Physical Systems `http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/78-ict-01-2014.html` (aufgerufen am 25.05.2014)

Literaturverzeichnis

Ackermann, E. (1996). Perspective-taking and object construction: two keys to learning. In Y. Kafai & M. Resnick (Hrsg.), *Constructionism in practice: designing, thinking, and learning in a digital world* (S. 25–35). Mahwah, NJ: Lawrence Erlbaum.

Ackermann, E. (2004). Constructing knowledge and transforming the world. In M. Tokoro & L. Steels (Hrsg.), *A learning zone of one's own: Sharing representations and flow in collaborative learning environments* (Bd. 1, S. 15–37.). Amsterdam, Berlin. Oxford, Tokyo, Washington, DC: IOS Press.

Aicher, O. (1991). *die welt als entwurf: schriften zum design* (1. Auflage Aufl.). Berlin: Ernst & Sohn.

Anderson, C. (2013). *Makers: das Internet der Dinge: die nächste industrielle Revolution*. München: Hanser.

Andriole, S. J. (1989). *Storyboard prototyping: a new approach to user requirements analysis*. Wellesley, Mass.: QED Information Sciences.

Bannon, L. (1991). From human factors to human actors: The role of psychology and human-computer interaction studies in system design. *Design at work: Cooperative design of computer systems*, 25–44.

Bean, J. & Rosner, D. (2014). Making: movement or brand? *interactions*, *21* (1), 26–27.

Beck, K. (2003). *Extreme programming*. München [u.a.]: Addison-Wesley.

Bellamy, R., Desmond, M., Martino, J., Matchen, P., Ossher, H., Richards, J. et al. (2011). Sketching tools for ideation. In (S. 808). New York, NY, USA: ACM Press.

Beyer, D. & Hassan, A. E. (2006). Evolution storyboards: Visualization of software structure dynamics. In *14th IEEE international conference on program comprehension, 2006. ICPC 2006* (S. 248–251). IEEE.

Binder, T. (1999). Setting the stage for improvised video scenarios. In *Chi '99 extended abstracts on human factors in computing systems* (S. 230–231). New York, NY, USA: ACM. Verfügbar unter `http://doi.acm.org/10.1145/632716.632859`

Blikstein, P. (2013). Gears of our childhood: Constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th international conference on interaction design and children* (S. 173–182). New York, NY, USA: ACM.

Bodker, S., Mathiasen, N. & Petersen, M. G. (2012). Modeling is not the answer! *interactions*, *19* (5), 54.

Bonanni, L. & Ishii, H. (2009). Stop-motion prototyping for tangible interfaces. In *Proceedings of the 3rd international conference on tangible and embedded interaction* (S. 315–316). New York, NY, USA: ACM.

Boujut, J.-F. & Blanco, E. (2003). Intermediary objects as a means to foster co-operation in engineering design. *Comput. Supported Coop. Work*, *12* (2), 205–219.

Brandt, E. (2007). How tangible mock-ups support design collaboration. *Knowledge, Technology & Policy*, *20* (3), 179–192.

Brandt, E. & Grunnet, C. (2000). Evoking the future: Drama and props in user centered design. In *Proceedings of participatory design conference (PDC 2000)* (S. 11–20). Verfügbar unter `http://www.itu.dk/courses/I/F2004/PDC00-drama-endelig.pdf`

Brereton, M. & McGarry, B. (2000). An observational study of how objects support engineering design thinking and communication: Implications for the design of tangible media. In *Proceedings of the SIGCHI conference on human factors in computing systems* (S. 217–224). New York, NY, USA: ACM.

Bruns, F. W. (2000). Complex objects and anthropocentric systems design. In L. Camarinha-Matos, H. Afsarmanesh & H.-H. Erbe (Hrsg.), *Advances in networked enterprises* (Bd. 53, S. 249-258). New York:

Springer.

Buchenau, M. & Suri, J. F. (2000). Experience prototyping. In *Proceedings of the 3rd conference on designing interactive systems: processes, practices, methods, and techniques* (S. 424–433). New York, NY, USA: ACM.

Budde, R. & Züllighoven, H. (1990). Prototyping revisited. In *CompEuro '90. Proceedings of the 1990 IEEE international conference on computer systems and software engineering* (S. 418–427).

Buechley, L., Eisenberg, M., Catchen, J. & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems* (S. 423–432). New York, NY, USA: ACM.

Buechley, L. & Hill, B. M. (2010). LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. In *Proceedings of the 8th ACM conference on designing interactive systems* (S. 199–207). New York, NY, USA: ACM.

Buur, J. & Andreasen, M. M. (1989, Juli). Design models in mechatronic product development. *Design Studies*, *10* (3), 155–162.

Bødker, S. (2006). When second wave HCI meets third wave challenges. In *Proceedings of the 4th nordic conference on human-computer interaction: changing roles* (S. 1–8). New York, NY, USA: ACM.

Bødker, S., Ehn, P., Sjögren, D. & Sundblad, Y. (2000). Co-operative design—perspectives on 20 years with 'the scandinavian IT design model'. In *Proceedings of NordiCHI* (Bd. 2000, S. 22–24). Stockholm. Verfügbar unter `http://cid.nada.kth.se/pdf/cid_104.pdf`

Carroll, J. M. (1997). Scenario-based design. In M. Helander, T. K. Landauer & P. V. Prabhu (Hrsg.), *Handbook of human-computer interaction* (2. Aufl., S. 383–406). Amsterdam: Elsevier.

Carroll, J. M. (2013). Human computer interaction (HCI). In M. Soegaard & R. F. Dam (Hrsg.), *The encyclopedia of human-computer interaction, 2nd ed.* Aarhus, Denmark: The Interaction Design Foundation. Verfügbar unter `https://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html`

Cilella, S., Berman, C. & Rheinfrank, J. (2010). Experience definition through storyboarding. In (S. 325). New York, NY, USA: ACM.

Confrey, J. (2006). The evolution of design studies as methodology. In R. K. Sawyer (Hrsg.), *The cambridge handbook of the learning sciences* (S. 135–152). Cambridge: Cambridge University Press.

Crilly, N., Maier, A. & Clarkson, P. J. (2008). Representing artefacts as media: Modelling the relationship between designer intent and consumer experience. *International Journal of Design*, *2* (3), 15–27.

Cross, N. (2001). Designerly ways of knowing: Design discipline versus design science. *Design Issues*, *17* (3), 49–55.

De Roeck, D., Slegers, K., Criel, J., Godon, M., Claeys, L., Kilpi, K. et al. (2012). I would DiYSE for it!: a manifesto for do-it-yourself internet-of-things creation. In *Proceedings of the 7th nordic conference on human-computer interaction: Making sense through design* (S. 170–179). New York, NY, USA: ACM.

Dewey, J. (1966). *Democracy and education: an introduction to the philosophy of education*. New York [u.a.]: Free Press.

Dittert, N. (2015). *TechSportiv. Technologie-Konstruktion: Ein „Gegenstand-mit-dem-man-denkt" für menschliche Bewegung*. Universität Bremen: Online-Veröffentlichung SUUB. (Dissertation)

Dittert, N., Katterfeldt, E.-S. & Reichel, M. (2012). TechKreativ: Tangible Interfaces in Lernwelten. In B. Robben & H. Schelhowe (Hrsg.), *Be-greifbare Interaktionen.* Bielefeld: Transcript.

Dittert, N., Katterfeldt, E.-S. & Schelhowe, H. (2012). Die EduWear-Umgebung – wearables konstruierend be-greifen. *i-com*, *11* (2), 37–43.

Dittert, N., Wajda, K. & Schelhowe, H. (2015). *Kreative Zugänge zur Informatik: Praxis und Evaluation von Technologie-Workshops für junge Menschen*. n.n. (Im Erscheinen)

Dougherty, D. (2012). The maker movement. *innovations*, *7* (3), 11–14.

Driskell, N. & Bardwell, M. (2013). *We are makers.* Verfügbar unter `http://vimeo.com/66162292` (Film)

Dubberly, H. (2009). ON MODELING: models of models. *interactions*, *16* (3), 54–60.

Dubberly, H. & Evenson, S. (2011). Design as learning—or "knowledge creation–the SECI model. *interactions*, *18* (1), 75–79.

Ehn, P. (2008). Participation in design things. In *Proceedings of the tenth anniversary conference on participatory design 2008* (S. 92–101). Indianapolis, IN, USA: Indiana University.

Erlhoff, M. & Marshall, T. (2008). Design. In M. Erlhoff & T. Marshall (Hrsg.), *Design dictionary* (S. 104–109). Basel: Birkhäuser.

Fallman, D. & Moussette, C. (2011). Sketching with stop motion animation. *interactions*, *18*, 57–61.

Faust, M. (2008). *Multi-Perspektivität in Modellierung und Simulation*. Universität Bremen: Online-Veröffentlichung SUUB. (Dissertation)

Floyd, C. & Klischewski, R. (1998). Modellierung - ein Handgriff zur Wirklichkeit. Zur sozialen Konstruktion und Wirksamkeit von Informatik-Modellen. In K. Pohl, A. Schürr & G. Vossen (Hrsg.), *Modellierung '98, Proceedings des GI-Workshops in Münster* (Bd. 9, S. 21–26). CEUR-WS.org. (11.-13. März 1998)

Floyd, C. & Züllighoven, H. (1999). Softwaretechnik. In P. Rechenberg & G. Pomberger (Hrsg.), *Informatik-Handbuch* (2. Aufl., S. 763–790). München, Wien: Hanser.

Flusser, V. (1993). *Vom Stand der Dinge: eine kleine Philosophie des Designs*. Göttingen: Steidl Gerhard Verlag.

Frauenfelder, M. (2011). *Made by hand: My adventures in the world of do-it-yourself*. New York [u.a.]: Portfolio Penguin.

Gänshirt, C. (2011). *Werkzeuge für Ideen: Einführung ins architektonische Entwerfen*. Berlin [u.a.]: Walter de Gruyter.

Gauntlett, D. (2011). *Making is connecting*. Cambridge [u.a.]: Polity.

Gaver, B. & Bowers, J. (2012). Annotated portfolios. *interactions*, *19* (4), 40–49.

Gershenfeld, N. A. (2007). *Fab: the coming revolution on your desktop–from personal computers to personal fabrication*. New York, NY, USA: Basic Books.

Gloger, B. (2013). *Scrum: Produkte zuverlässig und schnell entwickeln* (4., überarbeitete Aufl.). München: Carl Hanser Verlag GmbH & Co. KG.

Goorhuis, H. (1994). *Konstruktivistische Modellbildung in der Informatik*. Universität Zürich. (Dissertation)

Greenbaum, J. M. & Kyng, M. (1991). *Design at work: cooperative design of computer systems*. Hillsdale, NJ [u.a.]: Erlbaum.

Greenberg, S., Carpendale, S., Marquardt, N. & Buxton, B. (2012). The narrative storyboard: telling a story about use and context over time. *interactions*, *19* (1), 64–69.

Greenberg, S. & Fitchett, C. (2001). Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on user interface software and technology* (S. 209–218). New York, NY, USA: ACM.

Grube, G. (1995). Modellierung in der Informatik. *Abbild oder Konstruktion–Modellierungsperspektiven in der Informatik* (125), 3–20.

Hampel, T., Magenheim, J. & Schulte, C. (1999). Dekonstruktion von Informatiksystemen als Unterrichtsmethode – Zugang zu objektorientierten Sichtweisen im Informatikunterricht. In A. Schwill (Hrsg.), *INFOS '99: Informatik und Schule, 8. GI-Fachtagung Informatik und Schule* (S. 149–164). Berlin [u.a.]: Springer.

Hanser, E. (2010). *Agile Prozesse: Von XP über Scrum bis MAP* (1. Aufl.). München: Springer, Berlin.

Hartmann, B., Abdulla, L., Mittal, M. & Klemmer, S. R. (2007). Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on human factors in computing systems* (S. 145–154). New York, NY, USA: ACM.

Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A. et al. (2006). Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on user interface software and technology* (S. 299–308). New York, NY, USA: ACM.

Hashagen, A., Büching, C. & Schelhowe, H. (2009). Learning abstract concepts through bodily engagement: A comparative, qualitative study. In *Proceedings of the 8th international conference on interaction design and children* (S. 234–237). New York, NY, USA: ACM.

Hernandez, M., Calvo, R. & Alejos, R. (2010). *Arduino the documentary.* Verfügbar unter `http://vimeo.com/18539129` (Film)

Hesse, W. (2006). Modelle – Janusköpfe der Software-Entwicklung - oder: Mit Janus von der A- zur S-Klasse. In H. C. Mayr & R. Breu (Hrsg.), *Modellierung 2006* (Bd. 82, S. 99–113). Bonn: GI.

Hesse, W. & Mayr, H. C. (2008). Modellierung in der Softwaretechnik: eine Bestandsaufnahme. *Informatik-Spektrum*, *31* (5), 377–393.

Hochkeppel, W. (1989). Pragmatismus. In H. Seiffert & G. Radnitzky (Hrsg.), *Handlexikon zur Wissenschaftstheorie* (S. 270–274). München: Ehrenwirth.

Holbrook, H. (1990). A scenario-based methodology for conducting requirements elicitation. *SIGSOFT Softw. Eng. Notes*, *15* (1), 95–104.

Hornecker, E. (2011, März). The role of physicality in tangible and embodied interactions. *interactions*, *18*, 19–23.

Hornecker, E. & Buur, J. (2006). Getting a grip on tangible interaction: a framework on physical space and social interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems* (S. 437–446). New York, NY, USA: ACM.

Hornecker, E., Robben, B. & Bruns, F.-W. (2001). Technische Spielräume: Gegenständliche Computerschnittstellen als Werkzeug für erfahrungsorientiertes, kooperatives Modellieren. In I. Matuschek, A. Henninger & F. Kleemann (Hrsg.), *Neue Medien im Arbeitsalltag* (S. 193–216). Wiesbaden: Westdeutscher Verlag.

Hubwieser, P. (2004). *Didaktik der Informatik* (2. Aufl.). Berlin: Springer.

Humbert, L. (2002). Informatik – übergreifende, einzigartige Metawissenschaft? Überlegungen und fachdidaktischer Kontext. In S. E. Schubert, J. Magenheim, P. Hubwieser & T. Brinda (Hrsg.), *Forschungsbeiträge zur Didaktik der Informatik, Tagungsband des 1. Workshops der GI-Fachgruppe „Didaktik der Informatik" (DDI'02)* (Bd. 22, S. 109–118). Bonn: GI.

Iacucci, G., Iacucci, C. & Kuutti, K. (2002). Imagining and experiencing in design, the role of performances. In *Proceedings of the second nordic conference on human-computer interaction* (S. 167–176). New York, NY, USA: ACM.

Jonas, W. (1999). On the foundations of a science of the artificial. In *Useful and critical: The position of research in design international conference.* Helsinki. Verfügbar unter `http://home.snafu.de/jonasw/JONAS4-49.html`

Jonas, W. (2004). Forschung durch Design. *Erstes Design Forschungssymposium*. Verfügbar unter `http://8149.website.snafu.de/wordpress/wp-content/uploads/2011/08/2004_Basel.pdf`

Jung, M. F., Martelaro, N., Hoster, H. & Nass, C. (2014). Participatory materials: Having a reflective conversation with an artifact in the making. In *Proceedings of the 2014 conference on designing interactive systems* (S. 25–34). New York, NY, USA: ACM.

Kaschek, R. (1999). Was sind eigentlich Modelle? *EMISA Forum*, *9* (1), 31–35. Verfügbar unter `http://subs.emis.de/LNI/EMISA-Forum/Volume19_1/Was.sind.eigentlich.Modelle.pdf`

Kaschek, R. (2000). Schwachstellen einer Analyse des Modellbegriffs. In *EMISA FORUM* (Bd. 2000). Verfügbar unter `http://subs.emis.de/LNI/EMISA-Forum/Volume20_1/kaschek.pdf`

Katterfeldt, E.-S. (2007). *Conception and development of a modelling tool for children to support their software design processes.* Universität Bremen. (Unveröffentlichte Master Thesis)

Katterfeldt, E.-S., Dittert, N. & Schelhowe, H. (2009). EduWear: smart textiles as ways of relating computing technology to everyday life. In *Proceedings of the 8th international conference on interaction design and children* (S. 9–17). New York, NY, USA: ACM.

Katterfeldt, E.-S. & Schelhowe, H. (2008). A modelling tool to support children making their ideas work. In *IDC '08: Proceedings of the 7th international conference on interaction design and children* (S. 218–225). New York, NY, USA: ACM.

Katterfeldt, E.-S., Zeising, A. & Schelhowe, H. (2012). Designing digital media for teen-aged apprentices: a participatory approach. In *Proceedings of the 11th international conference on interaction design and children* (S. 196–199). New York, NY, USA: ACM.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.*, *37* (2), 83–137.

Knörig, A., Wettach, R. & Cohen, J. (2009). Fritzing: A tool for advancing electronic prototyping for designers. In *Proceedings of the 3rd international conference on tangible and embedded interaction* (S. 351–358). New York, NY, USA: ACM.

Koskinen, I., Zimmerman, J., Binder, T., Redstrom, J. & Wensveen, S. (2011). *Design research through practice: From the lab, field, and showroom* (1. Aufl.). Waltham, MA: Morgan Kaufmann.

Kühn, R., Keller, C. & Schlegel, T. (2011). Von modellbasierten Storyboards zu kontextsensitiven Interaction-Cases. *i-com*, *10* (3), 12–18.

Kurz, M. (2008). Model. In M. Erlhoff & T. Marshall (Hrsg.), *Design dictionary* (S. 261–262). Basel: Birkhäuser.

Kuutti, K., Iacucci, G. & Iacucci, C. (2002). Acting to know: Improving creativity in the design of mobile services by using performances. In *Proceedings of the 4th conference on creativity & cognition* (S. 95–102). New York, NY, USA: ACM.

Kuznetsov, S. & Paulos, E. (2010). Rise of the expert amateur: DIY projects, communities, and cultures. In *Proceedings of the 6th nordic conference on human-computer interaction: Extending boundaries* (S. 295–304). New York, NY, USA: ACM.

Lawson, B. R. (1979). Cognitive strategies in architectural design. *Ergonomics*, *22* (1), 59–68.

Lazar, J., Feng, J. H. & Hochheiser, H. (2010). *Research methods in human-computer interaction*. Chichester, U.K.: Wiley.

Lee, J. C., Avrahami, D., Hudson, S. E., Forlizzi, J., Dietz, P. H. & Leigh, D. (2004). The calder toolkit: wired and wireless components for rapidly prototyping interactive devices. In *Proceedings of the 5th conference on designing interactive systems: processes, practices, methods, and techniques* (S. 167–175). Cambridge, MA, USA: ACM.

Lim, Y.-K., Stolterman, E. & Tenenberg, J. (2008). The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, *15* (2), 1–27.

Löwgren, J. (2013). Interaction design. In M. Soegaard & R. F. Dam (Hrsg.), *Encyclopedia of human-computer interaction, 2nd ed.* Aarhus, Denmark: The Interaction Design Foundation. Verfügbar unter `http://www.interaction-design.org/encyclopedia/interaction_design.html`

Löwgren, J. & Stolterman, E. (2004). *Thoughtful interaction design: A design perspective on information technology*. Cambridge, MA [u.a.]: MIT Press.

Lunenfeld, P. (2000). Unfinished business. In *The digital dialectic: New essays on new media* (S. 6–22). Cambridge, MA [u.a.]: MIT Press.

Lüders, C. (2008). Beobachten im Feld und Ethnographie. In E. v. Kardorff, I. Steinke & U. Flick (Hrsg.), *Qualitative Forschung: ein Handbuch* (S. 384–401). Reinbek bei Hamburg: Rowohlt Taschenbuch-Verl.

Macaulay, C., Jacucci, G., O'Neill, S., Kankaineen, T. & Simpson, M. (2006). The emerging roles of performance within HCI and interaction design. *Interacting with Computers*, *18* (5), 942–955.

Magenheim, J. (2001). Deconstruction of socio-technical information systems with virtual exploration environments as a method of teaching informatics. In C. Montgomerie & J. Viteli (Hrsg.), *Proceedings of ED-MEDIA 2001, world conference on educational multimedia, hypermedia and telecommunications* (S. 1199–1204). Norfolk, VA: AACE.

Mareis, C. (2011). *Design als Wissenskultur: Interferenzen zwischen Design- und Wissensdiskursen seit 1960*. Bielefeld: Transcript.

Martinez, S. L. & Stager, G. S. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom*. Torrance, CA, USA: Constructing Modern Knowledge Press.

Mayring, P. (2002). *Einführung in die qualitative Sozialforschung: eine Anleitung zu qualitativem Denken* (5., überarb. und neu ausgestattete Aufl.). Weinheim [u.a.]: Beltz-Verl.

Mayring, P. (2010). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (11., aktualisierte und überarb. Aufl.). Weinheim [u.a.]: Beltz.

McCarthy, J. & Wright, P. (2004). *Technology as experience*. Cambridge, MA [u.a.]: MIT Press.

Millner, A. & Baafi, E. (2011). Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. In *Proceedings of the 10th international conference on interaction design and children* (S. 250–253). New York, NY, USA: ACM.

Mota, C. (2011). The rise of personal fabrication. In *Proceedings of the 8th ACM conference on creativity and cognition* (S. 279–288). New York, NY, USA: ACM.

Müller, D. (1998). *Simulation und Erfahrung: ein Beitrag zur Konzeption und Gestaltung rechnergestützter Simulatoren für die technische Bildung*. Universität Bremen. (Dissertation)

Müller, M. G. (2003). *Grundlagen der visuellen Kommunikation: Theorieansätze und Analysemethoden*. Konstanz: UVK-Verlag-Ges.

Nake, F. (2003). Informatik als Gestaltungswissenschaft: Eine Herausforderung an das Design. In K. H. Rödiger (Hrsg.), *Algorithmik - Kunst - Semiotik* (S. 142–163). Heidelberg: Synchron.

Nelles, W., Rhode, T. & Stechert, P. (2010). Entwicklung eines Kompetenzrahmenmodells – Informatisches Modellieren und Systemverständnis. *Informatik-Spektrum*, *33* (1), 45–53.

Norman, D. A. (1990). *The design of everyday things*. New York [u.a.]: Doubleday/Currency. (Orig. publ. u.d.T.: The psychology of everyday things. New York: Basic Books, 1988)

Obrenović, Ž. (2011). Design-based research: What we learn when we engage in design of interactive systems. *interactions*, *18* (5), 56–59.

Obrenović, Ž. & Martens, J.-B. (2011). Sketching interactive systems with Sketchify. *ACM Transactions on Computer-Human Interaction*, *18*, 1–38.

O'Sullivan, D. & Igoe, T. (2004). *Physical computing: Sensing and controlling the physical world with computers*. Boston, MA, USA: Cengage Learning.

Oulasvirta, A., Kurvinen, E. & Kankainen, T. (2003). Understanding contexts by being there: case studies in bodystorming. *Personal and Ubiquitous Computing*, *7* (2), 125–134.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York, NY, USA: Basic Books.

Paulos, E. (2012). You amateur! *interactions*, *19* (1), 52–57.

Pepper, P. (1992). *Grundlagen der Informatik*. München u.a.: Oldenbourg Wissenschaftsverlag.

Petri, C. A. (1977). Modelling as a communication discipline. In *Proceedings of the third international symposium on measuring, modelling and evaluating computer systems* (S. 435–449). Amsterdam: North-Holland Publishing Co.

Priese, L. & Wimmel, H. (2008). *Petri-Netze*. Berlin, Heidelberg: Springer.

Rechenberg, P. (1999). Formale Sprachen und Automaten. In P. Rechenberg & G. Pomberger (Hrsg.), *Informatik-Handbuch* (2. Aufl., S. 89–110). München, Wien: Hanser.

Regli, W. C., Hu, X., Atwood, M. & Sun, W. (2000). A survey of design rationale systems: approaches, representation, capture and retrieval. *Engineering with computers*, *16* (3-4), 209–235.

Reichel, M. (2008). *Tagging and smart textiles. A contextual perspective on constructionist learning environments*. Universität Bremen: Online-Veröffentlichung SUUB. (Dissertation)

Reichel, M., Osterloh, A., Katterfeldt, E.-S., Butler, D. & Schelhowe, H. (2008). EduWear: designing smart textiles for playful learning. In *Readings in education and technology: Proceedings of ICICTE* (S. 252–263). Corfu, Greece.

Resnick, M., Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T. et al. (2005). *Design principles for tools to support creative thinking* (Working Paper. Nr. Paper 816). Pittsburgh, PA, USA: Institute for Software Research. Carnegie Mellon University. Verfügbar unter `http://repository.cmu.edu/isr/816`

Resnick, M. & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *IDC '05: Proceeding of the 2005 conference on interaction design and children* (S. 117–122). New York, NY, USA: ACM.

Rosson, M. B. & Carroll, J. M. (2002). Scenario-based usability engineering. In *DIS '02: Proceedings of the conference on designing interactive systems* (S. 413–413). New York, NY, USA: ACM.

Rudd, J., Stern, K. & Isensee, S. (1996). Low vs. high-fidelity prototyping debate. *interactions*, *3* (1), 76–85.

Sanders, E. B.-N., Brandt, E. & Binder, T. (2010). A framework for organizing the tools and techniques of participatory design. In *Proceedings of the 11th biennial participatory design conference* (S. 195–198). New York, NY, USA: ACM.

Schelhowe, H. (1997). *Das Medium aus der Maschine: Zur Metamorphose des Computers*. Frankfurt/Main [u.a.]: Campus.

Schelhowe, H. (2007). *Technologie, Imagination und Lernen. Grundlagen für Bildungsprozesse mit Digitalen Medien*. Münster [et al.]: Waxmann.

Schelhowe, H. (2012). Interaktionsdesign für reflexive Erfahrung: Digitale Medien für Bildung. In B. Robben & H. Schelhowe (Hrsg.), *Be-greifbare Interaktionen* (S. 253–272). Bielefeld: Transcript.

Schinzel, B. (2001). Entwicklung eines Informationssystems für die Neurowissenschaften, Formale Modellierung am Beispiel der Hirntopographie. *Freiburger Universitätsblätter* (153), 33–50. Verfügbar unter http://www.careerbench.uni-freiburg.de/cms/fileadmin/publikationen/online-publikationen/formalemodellierung.pdf

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York, NY, USA: Basic Books.

Schön, D. A. (1992). Designing as reflective conversation with the materials of a design situation. *Research in Engineering Design*, *3* (3), 131–147.

Schubert, S. & Schwill, A. (2004). *Didaktik der Informatik*. Heidelberg, Berlin: Spektrum Akademischer Verlag.

Schütte, R. (1999). Zum Realitätsbezug von Informationsmodellen. In *EMISA forum* (Bd. 9, S. 26–36). Verfügbar unter http://subs.emis.de/LNI/EMISA-Forum/Volume19_2/emisa_99_2_Sch.pdf

Schäfer, K. & Bruns, W. (2001). PLC-programming by demonstration using graspable models. In *Proceedings of 6th IFAC symposium on cost oriented automation. 8.-9. 10. 2001.* Berlin. Verfügbar unter http://www.arteclab.uni-bremen.de/publications/artec-01-SchaeferBruns-PLCProgrammingbyDemo.pdf

Schütte, R. (2000). Realitätsbezug von Informationsmodellen – Detaillierte Erwiderung auf Kritik. Verfügbar unter http://www.pim.wiwi.uni-due.de/uploads/tx_itochairt3/publications/Arbeitsbericht_Nr._7.pdf

Seemann, J. & Gudenberg, J. W. v. (2000). *Software-Entwurf mit UML* (1. Aufl.). Heidelberg: Springer.

Sellen, A., Rogers, Y., Harper, R. & Rodden, T. (2009). Reflecting human values in the digital age. *Commun. ACM*, *52* (3), 58–66.

Shaer, O. & Hornecker, E. (2009). Tangible user interfaces: Past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction*, *3* (1-2), 1–137.

Sharples, M., McAndrew, P., Weller, M., Ferguson, R., FitzGerald, E., Hirst, T. et al. (2013). *Innovating pedagogy 2013: exploring new forms of teaching, learning and assessment, to guide educators and policy makers* (Bericht). Open University. Verfügbar unter http://www.open.ac.uk/personalpages/mike.sharples/Reports/Innovating_Pedagogy_report_2013.pdf (Open University Innovation Report 2)

Shipman, F. M. & McCall, R. J. (1997). Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, *11* (02), 141-154.

Shneiderman, B. (2008). Science 2.0. *Science*, *319* (5868), 1349–1350.

Singh, R. (2010). *Storyboard programming of data structure manipulations: a picture is worth 20 lines of code* [Thesis (M.Sc.)]. Verfügbar unter http://dspace.mit.edu/handle/1721.1/60187 (Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2010)

Spiessl, W., Villar, N., Gellersen, H. & Schmidt, A. (2007). Voodooflash: Authoring across physical and digital form. In *Proceedings of the 1st international conference on tangible and embedded interaction* (S. 97–100). New York, NY, USA: ACM. Verfügbar unter http://doi.acm.org/10.1145/1226969.1226989

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Wien - New York: Springer.

Stachowiak, H. (Hrsg.). (1983). *Modelle: Konstruktion der Wirklichkeit*. München: Fink.

Stachowiak, H. (1989a). Erkenntnistheorie, neopragmatische. In H. Seiffert & G. Radnitzky (Hrsg.), *Handlexikon zur Wissenschaftstheorie* (S. 64–68). München: Ehrenwirth.

Stachowiak, H. (1989b). Kybernetik. In H. Seiffert & G. Radnitzky (Hrsg.), *Handlexikon zur Wissenschaftstheorie* (S. 182–186). München: Ehrenwirth.

Star, S. L. & Griesemer, J. R. (1989). Institutional ecology,'translations' and boundary objects: Amateurs and professionals in Berkeley's museum of vertebrate zoology, 1907-39. *Social studies of science*, *19* (3), 387–420.

Thalheim, B. (2011). The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In D. W. Embley & B. Thalheim (Hrsg.), *Handbook of conceptual modeling* (S. 543–577). Berlin, Heidelberg: Springer.

Thomas, M. (2002). *Informatische Modellbildung. Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht.* Universität Potsdam. Verfügbar unter http://ddi.uni

-muenster.de/Personen/marco/Informatische_Modellbildung_Thomas_2002.pdf (Dissertation)

Torrey, C. & McDonald, D. W. (2007). How-to web pages. *Computer*, *40* (8), 96 –97.

Torrey, C., McDonald, D. W., Schilit, B. N. & Bly, S. (2007). How-to pages: Informal systems of expertise sharing. In *ECSCW'07: Proceedings of the tenth european conference on computer supported cooperative work* (S. 391–410). London: Springer.

Truong, K. N., Hayes, G. R. & Abowd, G. D. (2006). Storyboarding: An empirical determination of best practices and effective guidelines. In *Proceedings of the 6th conference on designing interactive systems* (S. 12–21). New York, NY, USA: ACM.

Turkle, S. & Papert, S. (1991). Epistemological pluralism and the revaluation of the concrete. In I. Harel & S. Papert (Hrsg.), *Constructionism* (S. 161–192). Norwood, NJ: Ablex.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA [et al.]: Harvard University Press.

Wahid, S., McCrickard, D. S., DeGol, J., Elias, N. & Harrison, S. (2011). Don't drop it!: Pick it up and storyboard. In *Proceedings of the SIGCHI conference on human factors in computing systems* (S. 1571–1580). New York, NY, USA: ACM.

Walter-Herrmann, J. & Büching, C. (Hrsg.). (2013). *FabLab–of machines, makers and inventors.* Bielefeld: Transcript.

Wedekind, H., Görz, G., Kötter, R. & Inhetveen, R. (1998). Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik. *Informatik Spektrum*, *21* (5), 265–272.

Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, *3* (3), 3–11.

Whitley, K. N. & Blackwell, A. F. (1997). Visual programming: The outlook from academia and industry. In *Papers presented at the seventh workshop on empirical studies of programmers* (S. 180–208). New York, NY, USA: ACM.

Williams, A., Gibb, A. & Weekly, D. (2012). Research with a hacker ethos. *interactions*, *19* (2), 14–19.

Winograd, T. (1997). The design of interaction. In *Beyond calculation: The next fifty years in computing* (S. 149–161). New York: Springer.

Wright, P. & McCarthy, J. (2008). Empathy and experience in HCI. In *Proceedings of the SIGCHI conference on human factors in computing systems* (S. 637–646). New York, NY, USA: ACM.

Anhang

A Analysekategorien

Kategorie	Definition	Ankerbeispiele (für textuelle Modelle)	Kodierregeln
Programm (Programmierung)	formalisierende Beschreibung des Artefaktverhaltens ähnlich Pseudocode oder in Programmiercode	"(...) Werte die höher sind als 350 (...) Der Abstandswarner schaltet den Vibrationsmotor (...) und die LEDs (...) an".	Programmcode, Algorithmus, Pseudocode, formale Werte. Viertel/halbe Wertung: enthält auch unformale Infos über Umwelt/Einbettung; Attribut/Verb in Zusammenhang mit unformalen Eigenschaften.
Hardware (Hardwarekonfiguration)	statische Hardwarekonstruktion des Artefakts	„11 - Display, 13 - LED“	Abbildungen von Hardware-Teilen und -Verbindungen dieser konkret benannt/erkennbar. Viertel/halbe Wertung: keine Anschlüsse; gemischt mit Gestaltbeschreibung.
Korpus (Gestalt des Artefaktkorpus)	das statische stoffliche Artefakt	„Der Kobold ist wie gesagt pink und sein Zauberstab leuchtet“	Fotos, Skizzen von Artefakten, die deren Gestalt abbilden; zeigt Form; bildet ggf. Umwelt/Kontext mit ab; keine/wenig formale Beschreibung. Viertel/halbe Wertung: benennt nur einzelne Eigenschaften (z.B. Farbe); gemischt mit Hardware (z.B. LEDs am Kopf statt Augen).
Interaktion (Interaktives Verhalten)	unformale Beschreibung des dynamischen Verhaltens bzw. Interaktion mit dem Artefakt	„Es ertönt ein nerviges Summen, das die Frau warnt und den Mann abschreckt“	bildet dynamisches Verhalten in Umwelt/Kontext ab; keine/wenig formale Beschreibung. Viertel/halbe Wertung: Andeutungen, nur einzelne Aspekte erkennbar.
Pragmatische Merkmale			
– Kontext	der Kontext, für den das Artefakt gemacht ist und/oder dem es entstammt.	„Der Schwimmohörus ist ein ganz besonderes Tier. Er kann nämlich mit seinen langen Ohren besonders gut hören. Er lebt in flachen Gewässern.“	beschreibt/zeigt ausführlich/gut erkennbar den Kontext des Artefakts. Viertel/halbe Wertung: angedeutet in Attributen o.ä.; auf Fotos nur einzelne Aspekte im Hintergrund erkennbar.
– Intention	der Zweck, für den das Artefakt gedacht ist und warum es erstellt werden soll	„Die Idee kam uns, weil unser Roboter halt was mit „Zauberei“ zu tun haben sollte, und wir dachten, dass das cool wäre, wenn die Augen ..“	Gründe werden eindeutig genannt/thematisiert. Viertel/halbe Wertung: angedeutet in Attributen o.ä.
– Modellsubjekt	die Erfinder_innen und Konstrukteur_innen des Artefakts	„Wir ([Namen der Autor_innen]) entwarfen und bauten den (...)“	Namen werden genannt; Personen auf Fotos abgebildet.
Graphische Modelle (und Unterkategorien, s. Abb. 3.1, S. 37)	zweidimensionale, visuelle deskriptive Darstellungen (+ Kriterien nach AMT für jeweilige Unterkategorien)	Ikonisches Bildmodell/Bild: Foto eines Artefakts. Ikonisches Bildmodell/teilschematische Abb.: Amici-Block mit Hardware-Icon.	erfüllt jeweilige Definition; auch mehrere Zuordnungen möglich; Fotografie von Skizze gilt als Skizze; Icons/Blöcke des Amici-Code werden nicht zu Bildmodellen in Projekt addiert sondern gesondert behandelt.
Textuelle Modelle (und Unterkategorien, s. Abb. 3.1, S. 37)	Zeichenmodelle mit lateinischen Sprachzeichen (+ Kriterien nach AMT für jeweilige Unterkategorien)	nicht-formale, umgangsspr. t. M.: beschreibender Text (z.B. „Der Schwimmohörus ist ein ganz besonderes Tier (...)“); formales, programmierspr. t. M.: text. Arduino-Code (z.B. „void setup() { Serial.begin(9600); (...)“)	erfüllt jeweilige Definition; auch mehrere Zuordnungen möglich; Programmcode wird nicht zu text. Modellen des Projekts addiert.

Technische Modelle (und Unterkategorien, s. Abb. 3.1, S. 37)	materiell und dreidimensional, raum-zeitlich oder materiell-energetisch (+ Kriterien nach AMT für jeweilige Unterkategorien)	kaum Vorkommen aufgrund gewählter Methoden/Tools, evtl. Physikot., elektrot. M. als Hardwareaufbau (auf Foto); (Physikot., mechanisches M. wäre z.B. Prototyp für Mechanik)	erfüllt Definition, d.h. ist dynamisch/raum-zeitlich oder materiell-energetisch, daher kein unmittelbares Vorkommen in vorliegenden Modellen.

Tabelle A.1: Kategorien und ihre Ausprägungen: Definitionen, Ankerbeispiele und Kodierregeln

B Überblick Modellanalyse

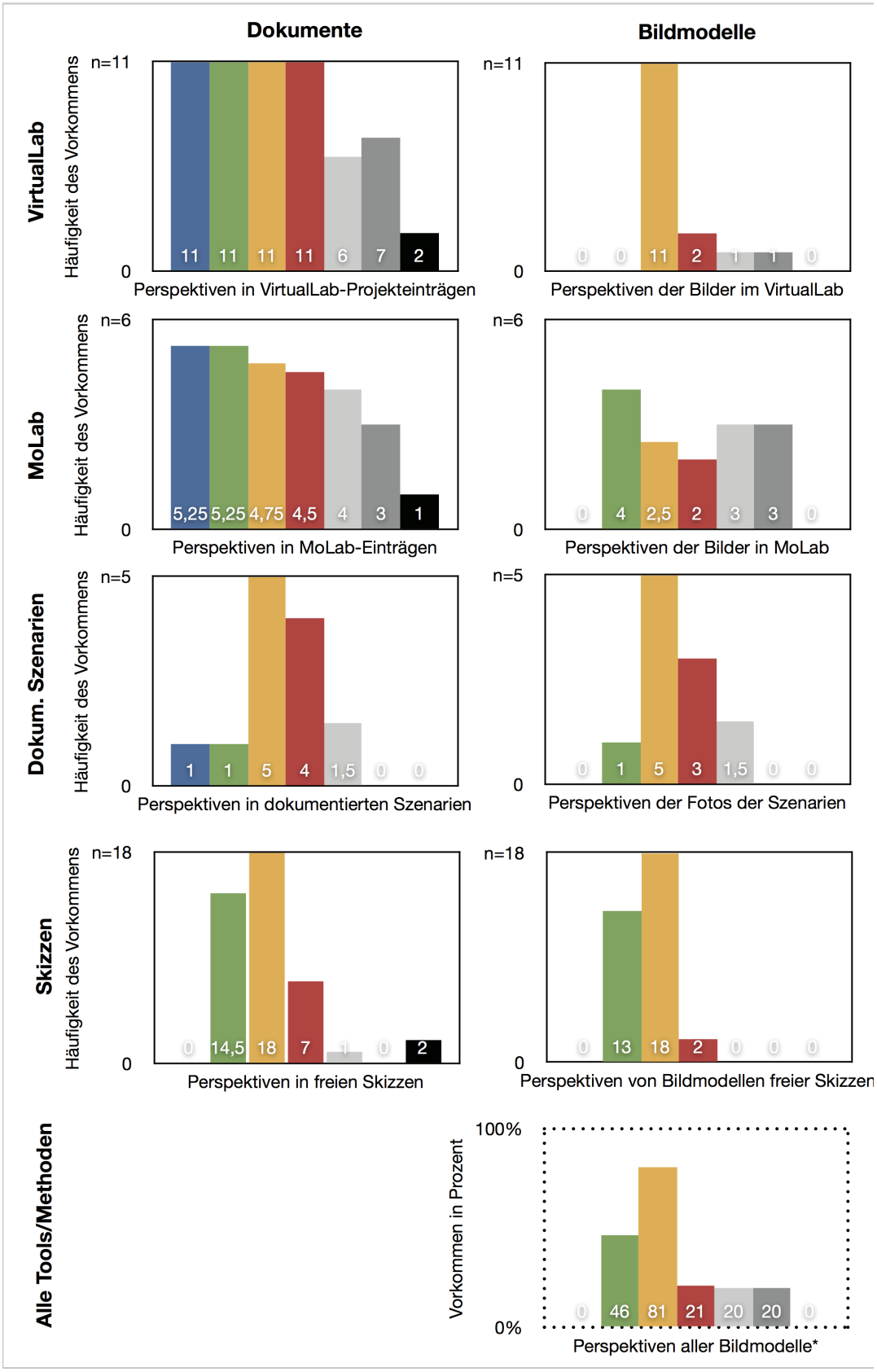
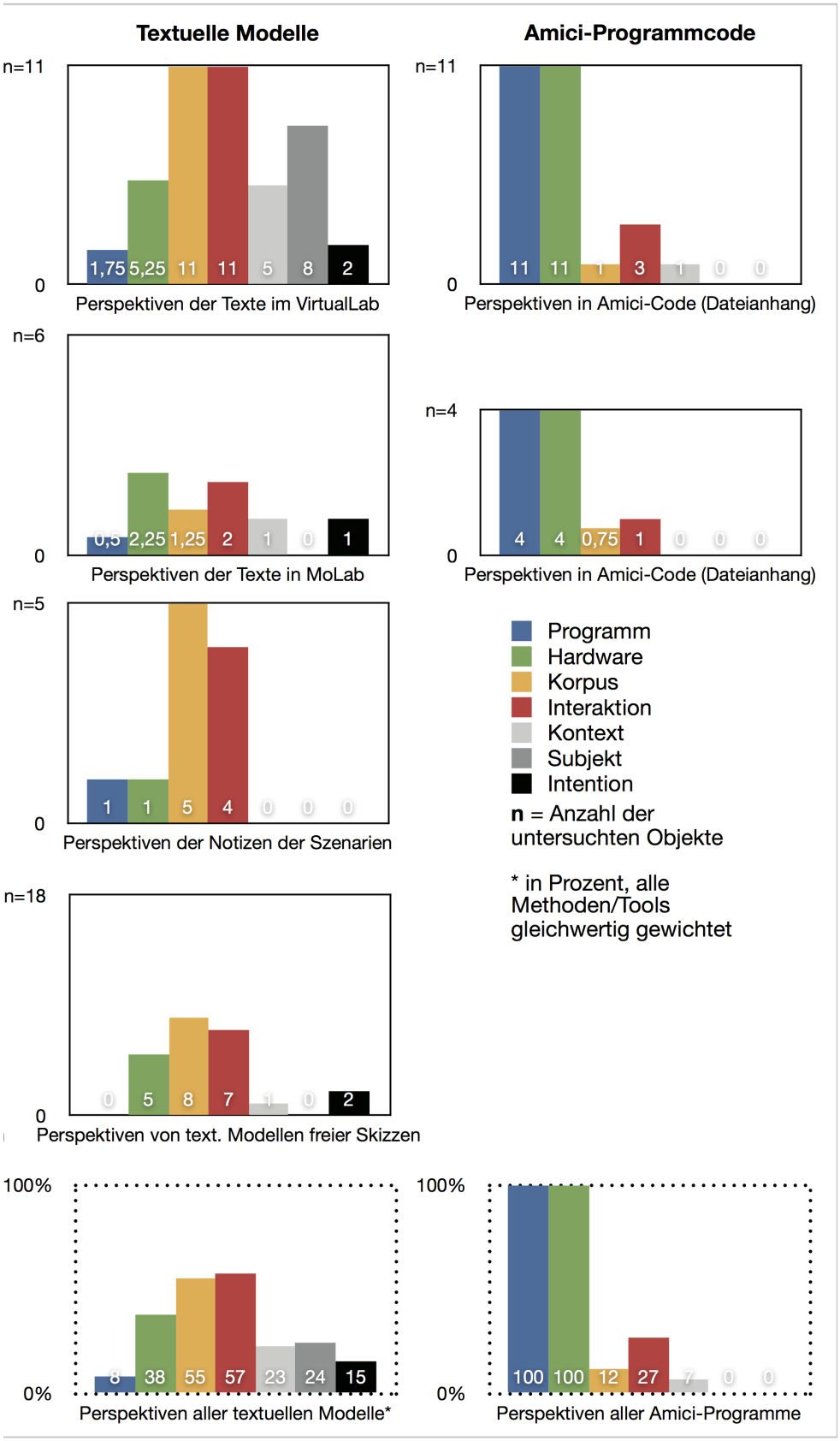


Abbildung B.1: Vergrößerte Darstellung der quantitativen Ergebnisse der Modellanalyse: Vorkommen von Perspektiven je Methode/Tool und Modellart


Textuelle Modelle
Amici-Programmcode
n=11
0
1,75 5,25 11 11 5 8 2
Perspektiven der Texte im VirtualLab
n=11
0
11 11 1 3 1 0 0
Perspektiven in Amici-Code (Dateianhang)
n=6
0
0,5 2,25 1,25 2 1 0 1
Perspektiven der Texte in MoLab
n=4
0
4 4 0,75 1 0 0 0
Perspektiven in Amici-Code (Dateianhang)
n=5
0
1 1 5 4 0 0 0
Perspektiven der Notizen der Szenarien
Programm
Hardware
Korpus
Interaktion
Kontext
Subjekt
Intention
n = Anzahl der untersuchten Objekte
* in Prozent, alle Methoden/Tools gleichwertig gewichtet
n=18
0
0 5 8 7 1 0 2
Perspektiven von text. Modellen freier Skizzen
100%
0%
8 38 55 57 23 24 15
Perspektiven aller textuellen Modelle*
100%
0%
100 100 12 27 7 0 0
Perspektiven aller Amici-Programme

C Beobachtungsleitfaden

Fragen des Beobachtungsleitfadens:

- Anzahl der Teilnehmenden und Geschlecht?
- Wo treten Probleme hinsichtlich der Bedienbarkeit auf?
- Wird das Konzept verstanden? Wann/wie/warum werden Modelle erstellt?
- Inwiefern werden vorherige Modelle wieder aufgegriffen oder spielten sie sonst eine Rolle?
- Treten unerwartete Vorkommnisse auf?

D Leitfaden Auswertungsgespräch

Dokumentierte Projekte werden gemeinsam auf Ausdrucken angeschaut.

Warum wurden gerade diese Elemente erstellt?
Was zeigen oder dokumentieren sie aus Sicht der Befragten?
Haben sie dabei geholfen, das Projekt in der nächsten Woche fortzuführen?
Hat die Anordnung auf der Übersichtsseite eine Bedeutung für die Befragten?

Die Teilnehmenden sollen sich vorstellen, sie würden zukünftig auch alleine ein Arduino-Projekt verfolgen.

Wie würden die Befragten dann ihre Projekte dokumentieren?
Würden sie die Elemente auf der Übersichtsseite anders anordnen?
Würden sie sich noch weitere Hilfsmittel wünschen?

Dot Voting: Die Teilnehmenden werden gebeten, je fünf farbige Klebepunkte für Funktionen oder Gestaltung, die gefallen (blau) bzw. nicht gefallen (schwarz) auf Ausdrucken der Plattform zu verteilen.

Diskussion mit Begründung der Entscheidungen und Verbesserungsvorschlägen.